Nubes: Towards building a Secure and Scalable Hybrid Cloud Infrastructure

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Milan Shetti

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy

David Hung Chang Du

April, 2021

Acknowledgements

There are many people I would like to acknowledge for their support and guidance during the period of completing my doctoral degree. First and foremost, I would like to thank my advisor Professor David Du. Pursuing Doctoral degree while holding a full time CTO role at the largest business in Hewlett Packard Enterprise was intense and truly gratifying on personal and professional level. I would not have been able to complete this journey without guidance and support from Professor David Du.

All the faculty members at University of Minnesota with whom I had the pleasure of inter-acting had been of tremendous help. At the University, I want to especially thank Associate Professor, Arindam Banerjee for been extremely patient with me while I brushed up and got re-acquainted with machine learning. Also special thanks to Professor Jon Weismann for accepting me as an independent study student to help complete my breadth requirements. Special thanks to Professor Abhishek Chandra and Ulya Karpuzcu for accepting to be on my thesis advising committee.

I would like to thank my parents for their support and sacrifices in my early years. Almost two decades after I completed by Masters, I am grateful to have the opportunity of completing my PhD. To my wife Susan and my boys Ethan and Sean, words don't begin to express how grateful I am to you all for the support, encouragement and motivation you gave me to complete this PhD. Susan, Ethan and Sean, I know all the sacrifices you had to endure – all my time away from you guys in my home office working on the research.

Special thanks go to all my team and mentors at Hewlett Packard Enterprise who provided me moral support to a crazy idea of pursing PhD while working as CTO of high pace multi-billion-dollar businesses. Hewlett Packard Enterprise will always have a special place in my heart.

I would also like to thank all the students at Center of Research of Intelligent Storage (CRIS) for providing me the spark to initiate joining the PhD program. Although all the interactions with students were rewarding, special thanks for Bingzhe Li and Zhichao Cao for their assistance. I always wanted to pursue my PhD, after graduating from Syracuse University with MS in Computer Engineering, I never imagined getting another opportunity to pursue my academic interest. Few years ago, when I first visited the University of Minnesota campus during the CRIS meeting and inter-acting with students and professors on industry-academic research it gave me the spark to re-start pursuit of PhD. I will be ever thankful for the CRIS program to be that catalyst for re-igniting the spark.

Dedication

I dedicate this thesis to my family; my wife Susan, my two sons Ethan and Sean and my parents for their unconditional love and support.

Abstract

Industry accepted definition of Hybrid Cloud is an infrastructure which spans Public Cloud, off premise from customer's data center and Private Cloud, on premise to the customer's data center. Public Cloud has sustainable economics of scale (cost) and ubiquitous easy access advantage over Private Cloud whereas, Private Cloud has security, privacy and predictable performance and availability advantage over Public Cloud. A Hybrid Cloud conceptually can combine the advantages of both Private and Public Cloud however, there are number of challenges especially with the Storage technologies to provide secure and scalable Hybrid Cloud infrastructure. In this thesis, we propose a framework to build secure and scalable hybrid cloud infrastructure.

With the advent of Server Virtualization, it is possible to move applications between the Private and Public Cloud. With the advent of Container technologies and Micro-Services based paradigm for application development it is possible to burst compute needs from private cloud to public on an on-demand basis. However, Storage infrastructure pose considerable technical challenges to realize the Hybrid Cloud vision in practice. There are two major issues with Storage in Hybrid Cloud: (1) Storage has Gravity and (2) Storage Protocols are inherently insecure.

In the first part of the thesis, we will first examine the issues with workload mobility. Application migrations or bursting within Hybrid Cloud is bottlenecked by the Storage infrastructure. It is not commercially viable to keep a mirrored copy of all data between the Private and Public clouds simultaneously to enable workload migration through Virtual Machine Migration or Containers Micro-Services. The amount of data which needs to be transferred between Private and Public cloud is too large. The simple access pattern based heuristic based model to determine the data to move between elements of Hybrid Cloud is computationally prohibitive. In order to address these storage migration challenges, we will propose machine learning (Support Vector Machine) based solution.

In the second part of the thesis, we will examine the known security vulnerabilities of each Storage protocols used in Hybrid Cloud, namely: a) Block Storage (iSCSI), b) File Protocol (NFS) and c) Object Protocol (S3). These storage protocols were designed as

simple point to point inter-connect technologies and in time haven't evolved beyond just the performance optimization. The protocols are susceptible to simple vulnerabilities such as man in the middle attacks and more. And in this part of the thesis, we will provide a new Storage Protocols paradigm using Location Based Services to enhance the security model for data access.

And finally, in the third part of the thesis, we propose a Secure and Scalable Hybrid Storage (SSHS) framework by combining the Machine Learning techniques for Storage Mobility and Location Based Services to enhance Security overcomes the major barriers in adoption and deployment of the Hybrid Cloud Infrastructure. The experimental results demonstrate the framework to self-learn and self-manage data mobility based on the workload in Hybrid Cloud and also demonstrates the power of integration of location-based services with the Storage protocol to secure chain of trust data access from Application to Storage.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The focus of the PhD Thesis is to propose and develop a) Cost Effective High Performance, b) Highly Scalable, and c) Secure Hybrid Cloud Infrastructure. The proposed framework, called Nubes (Latin word for Cloud) uses several combinations of machine learning techniques and location-based services to build highly scalable and secure Hybrid Cloud Infrastructure.

A Hybrid Cloud environment is an infrastructure environment where application or workload spans at least one Public cloud environment [1] [2] [3] and at least one private data center (Private cloud). Server virtualization technologies [4] [5] [6] have made such building Hybrid Cloud possible. The motivation for using the Public Cloud is cost benefits at smaller scale and flexibility of bursting on-demand workload spike should the Private Cloud runs out of performance or storage capacity. And Private Clouds are essentially the existing virtualized data center, which provide enterprises performance, inherent control and security.

The problem space in Hybrid Cloud environment is very large and poses unprecedented large scale resource optimization problems. Data Mobility and Data Security at Scale in a Hybrid Cloud environment is a relatively new research topic. And due to its scale, a big focus area of research is the application of machine learning techniques for the enterprise environment.

There are three fundamental elements to Nubes framework, each corresponding to its above-mentioned focus area. This chapter will provide a quick overview of the problem statement and solutions to each of these three elements.

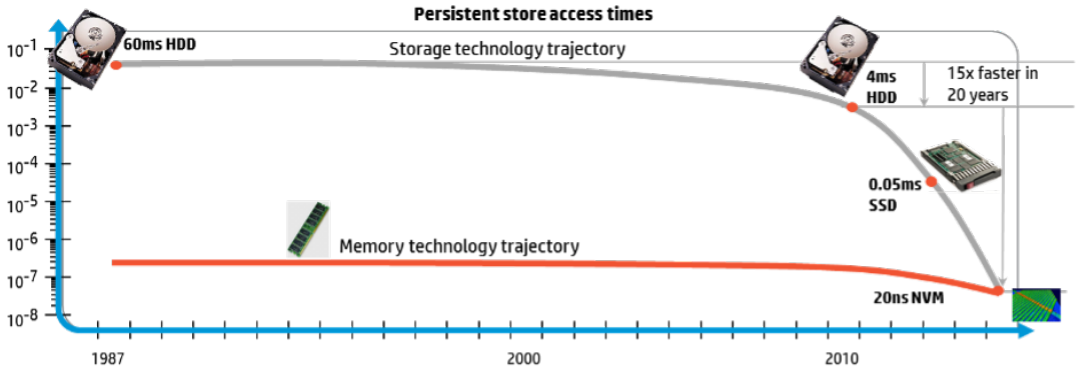

Figure 1.1: An illustration of the performance compares of these different media types. Courtesy: Hewlett Packard Enterprise

1.1 Local Performance Optimization for Hybrid Cloud

To take the best advantage of high performance (but expensive) solid state technologies, Nubes provides a novel data tiering technique using Support Vector Machine to tier the data to achieve the best price/performance characteristics for the Storage System.

This optimization technique can be applied to both the private cloud and the public cloud environment. In Chapter 2 we will study this optimization in detail applied to private cloud environment and compare the results with the latest state of art data tiering techniques in the Enterprise Storage industry.

The relatively recent commercialization of high-performance solid-state storage media technologies, such as Flash and Non-Volatile Memory [10] [11] is presenting new architectural considerations when introduced in the enterprise storage system.

Although the performance of these storage media are magnitude of orders greater than the hard disk drives, the cost of these new media on $/GB is still quite high. In order to balances the cost and performance in scalable enterprise storage system, a new class of storage systems have emerged, called Hybrid Storage Systems.

These Hybrid Storage Systems use a combination of low cost, low performance hard disk drives with a combination of higher cost, higher performance SSD drives and Persistent Memory. Figure 1.2 illustrates the topology of external storage.

Typically there are 100s of servers (each with 4 to 8 Virtual Machines) attached via either Ethernet Network or Fiber Channel Network. Attached to these networks are

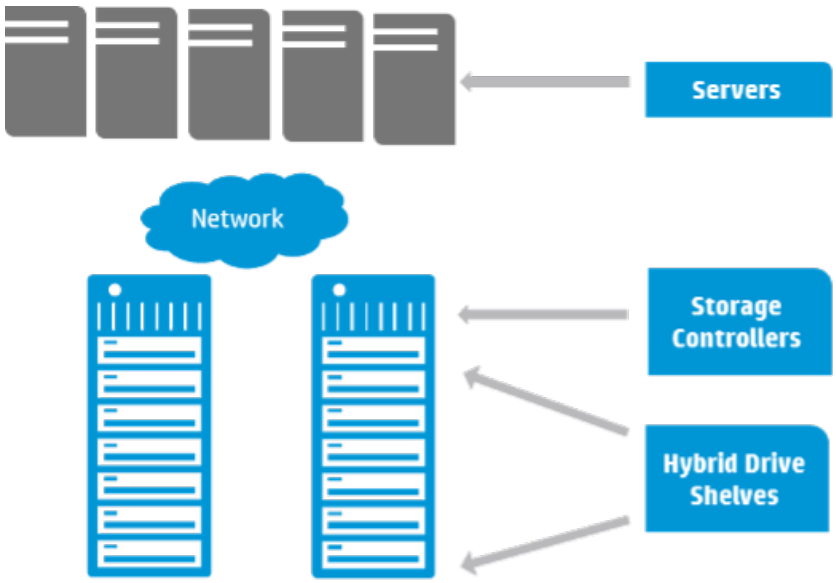

Figure 1.2: An external storage topology in the Private Cloud.

External Storage arrays, these Storage arrays have 2 to 8 node Storage processing units and shelves of storage media either hard disk drives or SSDs in each of these shelves.

In addition to serving the IO from drives to the Servers, these Storage controllers are responsible for all the data tiering within the external storage to provide the maximum IO from the fast tier media, SSD.

Problem Statement: These Storage Controllers are the key resource constraint in this architecture to optimize. These Storage Controllers are responsible for multiple functions in the system including providing IO to the Host applications, so its involvement in data tiering to move data from lower media to faster media competes in constraint resource envelope competes with the application performance.

Current Solution: The current solutions do not factor in a) an overall cost of migration, including the cost of latency impact to host during the data migration and b) aggressively migrate more data than they need to, eventually having to re-migrate the data in near subsequent iterations.

Nubes Novelty: Support Vector Machine are an excellent two class classifier supervised machine learning technique. By applying Support Vector Machine to the data tiering decision process, the system a) by maximizing the decision boundary in affect

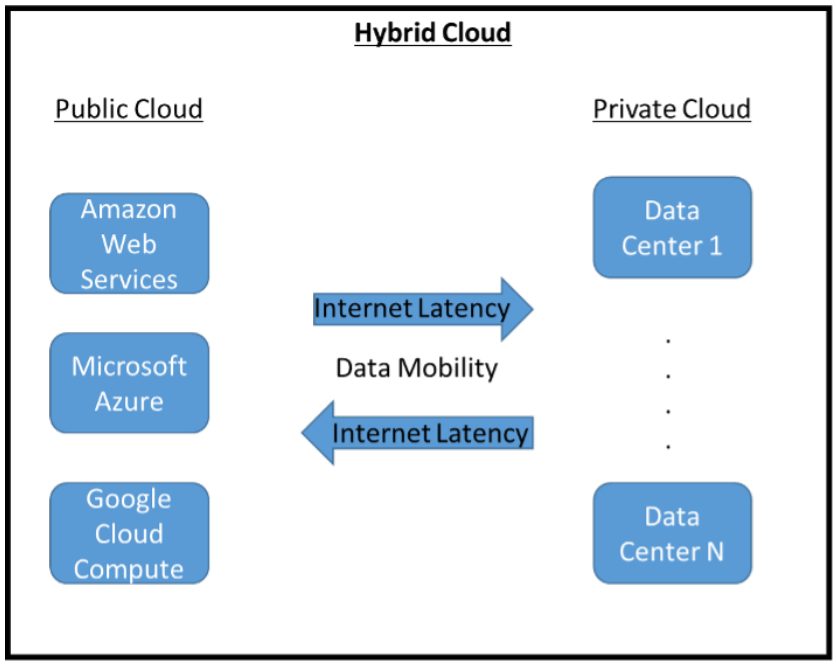

Figure 1.3: Hybrid Cloud, combination of Enterprise Data Center and Public Cloud.

factors the cost of migration between the two storage tiers (classes) and b) at every iteration recalculating, self learns the decision boundary factors by adding more support vectors provides an effective but lazy approach to data tiering.

1.2 Global Performance Optimization for Hybrid Cloud

Making sure the right virtual machine is in the appropriate cloud, private or public has impact on cost and performance of the Hybrid Cloud. It is not commercially practical to have two copies of data one if public cloud and one in private cloud for all the virtual machines in the data center.

Through the lifecycle of the application development and the virtual machine, the access pattern of the virtual machine can change over time. Moving data requires time and bandwidth. So pre-fetching virtual machine datastore based on probabilistic model of access pattern is very much desirable.

In Chapter 3, as part of Nubes framework we propose an algorithm to pre-fetch the datastore for the virtual machine in anticipation of virtual machine movement either

from private to public cloud and vice versa. The outcome from the novelty of this approach is that this greatly reduces the need for multiple copies for solely performance reason.

Problem Statement: As illustrated in figure 1.3, Hybrid Cloud is combination of the infrastructure at Enterprise Customers on premise infrastructure and one/or all of the top 3 public cloud providers. The workloads in Hybrid Cloud infrastructure are either virtualized or containerized. Should for either cost, control or performance regions the workload is to be moved between public and private cloud the large amount of the data transfer has to occur over the internet latency and not real time.

Current Solutions: In most cases today the customers are trapped at the origin of virtualized workload creation. In other words, if the virtual machine originates in public cloud the data stays on public cloud and should the virtual machine originate in private cloud (which is majority of customer data), then the data and the virtual machine stay in the private cloud. Some tiering or replication between private and public cloud are available but of limited use due to the prohibitive cost of having two copies of everything on each private and public cloud. In academic research, there have been proposals around hint-based data pre-fetch wherein application will provide the hint to the infrastructure on change its lifecycle and need to pre-fetch the data. The problem which is this approach is that it requires applications or protocols to change, which is possible for some but not all for all situations.

Nubes Novelty: Nubes applies machine learning techniques to pre-fetch the data as the solution. Fundamentally, Nubes leverages the property of machine learning technique, 'evidence updates belief'. With Nubes probabilistic model, we first assign high belief that the data for the virtual machine be on the same cloud where the virtual machine originated. But we update the belief based on the evidence on that the cost, performance or control attributes have changed for each virtual machines. Based on the probabilistic model, we pro-actively tier the data between the clouds. The experiment results are presented in Chapter 3.

1.3 Location aware data access for Storage Protocol in Hybrid Cloud

Ethernet based Storage protocols, iSCSI [7] and NFS [8] [9] were designed long before virtualization come to the commercial market. The goal of these storage protocols are to make server and client connect easily and they did have limited security abilities. Neither of these protocols were designed with Cloud in mind either. Both these protocols are susceptible to a number of security vulnerabilities.

In Chapter 4, we will first document the known vulnerabilities of key storage protocols. As part of Nubes framework we introduce a novel idea of integrating these protocols with Local Based Services [10] as a key authentication method for data access. In addition to improved authentication, the added advantage of this approach is that it provides an infrastructure for security framework whereby policies of state and countries can be enforced by the Hybrid Cloud infrastructure.

One of the concerns generally expressed with new authentication method on the overall performance of the system. To address the concern, as part of Nubes framework we will demonstrate performance optimization methods for using the Location Based Services with Hybrid cloud use case where some part of the data is located in public cloud and some part in private cloud.

Problem Statement: With the advent of virtualization, the server hosting the application essentially became mobile. In a hybrid cloud environment, the application can move from one data to other another in private cloud or also move between public and private cloud. This application mobility creates a new surface area of attack for data security and also creates challenges for enforcing security regulations and policies either imposed by the corporation or government agencies. Cloud storage protocols for virtual machines, such as iSCSI and NFS at present day do not have any specification areas focused on mobility aspect of virtualization.

Current Solutions: Most of the academic research and commercial solutions have been focused on the malware detection or the use of Location based Services for consumer applications.

Nubes Novelty: Nubes provides a security framework a) integrates iSCSI security framework with the Wireless Access Point inter-datacenter and intra-datacenter Private

Cloud security and b) integrates iSCSI security framework with the open-source Location Based Services for Hybrid Cloud, between Public and Private Cloud. The Nubes security framework is discussed in Chapter 4, we also discuss the proposal to improve the performance of the Security framework, so it doesn't become the bottleneck for the Hybrid Cloud infrastructure. This framework can also be extended to NFS and S3 storage protocols.

- Chapter 1 Introduces the various Enterprise IT trends, motivation, design challenges and research areas for Hybrid Cloud.
- Chapter 2 Focuses on the optimization of resources in Hybrid Storage System by proposing a resource efficient data mobility in a hybrid storage system.
- Chapter 3 Proposal for an elastic virtual machine scheduling algorithm to minimize the total cost of ownership in Hybrid Cloud is discussed.
- Chapter 4 Regulatory Compliance Considerations for Hybrid Cloud are introduced and reference framework to implement security model modernizing the storage protocol for security in hybrid cloud and regulatory compliance are discussed.
- Chapter 5 Provides conclusion for the thesis and discussion for future research work are laid out.

Chapter 2

Resource Efficient Data Mobility in Hybrid Storage Systems

2.1 Introduction

Most large-scale enterprise storage systems are designed as hybrid storage systems with two classes of storage media: a small number of higher performance expensive devices (performance tier) and a large number of lower performances high capacity inexpensive devices (capacity tier). The goal of such a system is to serve majority of the I/O requests from performance tier and store less frequently used data in capacity tier. Therefore, it often requires moving data between tiers. A large data migration volume between tiers can cause a huge overhead in practical hybrid storage systems. Therefore, how to balance the trade-off between the migration cost and potential performance gain is a challenging and critical issue in hybrid storage systems.

In this chapter, we focused on the data migration problem of hybrid storage systems with two classes of storage devices. A machine learning (ML) based migration algorithm called adaptive Support Vector Machine (SVM) migration algorithm is proposed. This algorithm is capable of more precisely classifying and efficiently migrating data between performance and capacity tiers. Moreover, this adaptive SVM migration algorithm involves K-Means clustering algorithm to dynamically select a proper training dataset such that the proposed algorithm can greatly reduce the volume of migrating data. Finally, the real implementation results indicate that the ML-based algorithm reduces

the migration data volume by about 40% and achieves 70% lower latency compared to other algorithms.

Unprecedented and ever-increasing 3 V's (Volume, Velocity and Variety) of data continues to put pressure on storage systems to find cost-effective solutions capable of delivering peak performance for all possible workloads [7]. Recently, different types of emerging storage devices come out [8, 9], which have different density and performance. For example, flash-based Solid State Drives (SSDs) can achieve much faster random access performance with low latency compared to traditional Hard Disk Drives (HDDs) while HDDs are much cheaper than SSDs. Therefore, it is not cost-effective to build a petabyte byte (PB) storage system using only fast devices [10]. Compared with different types of emerging devices, they can have a 100x latency difference and more than 5x price difference. These differences have motivated storage vendors to build two-level hybrid storage systems with different types of storage devices.

A key characteristic of data that remains unchanged is that data has an access life cycle (i.e., not all data are accessed at all times by applications). The desired outcome for a hybrid storage system is to deliver almost all the IO operations from a high-performance tier (e.g., SSDs). To achieve this desired outcome, data have to be moved between tiers depending on the frequency of IO accesses (a process is referred to as data migration). Although data migration between tiers introduces overheads, given a 100x $/IOPS difference between SSDs and HDDs, this also presents an opportunity to design and develop a migration algorithm which can be cost-effective and can also deliver peak performance as demanded by applications. Some previous studies have investigated hybrid storage systems [11, 12, 13, 14, 15]. They formulated the characteristics of workloads and the properties of devices based on statistical analysis. However, migration optimization has the complexity of NP-hard [11]. To avoid the difficulty of solving the NP-hard problem, those researchers simplified the problem and proposed polynomial time bound heuristic solutions. However, the simplified formulas are not able to precisely express the behaviors of workloads. As a result, the miss-expression may result in a large migration volume and decreasing the performance gain in a hybrid storage system. Machine learning (ML) as a classifier has been successfully used in many applications [16, 17]. It can be a good candidate to solve the data migration problem with less migration volume and higher performance gain. This is because the data migration in hybrid

storage systems can be regarded as a classification issue to determine/classify data to which storage tier they should reside.

In this chapter, we focus on a hybrid storage system containing two types of storage devices (e.g., SSD and HDD) and propose a K-Means assisted Support Vector Machine (K-SVM) migration algorithm. In this algorithm, time is partitioned into periodical duration. In each period, the request access patterns are collected. At the end of the current period, a K-SVM classifier is used based on the request access patterns of this period. Then, a classifier is used to determine which data should be migrated to a different tier in the following period. To increase the precision of the classifier, the K-Means clustering algorithm is introduced to dynamically select a proper training dataset such that the overall migration size can be reduced. Furthermore, we investigate the influence of different capacity ratio between two types of storage on the performance of the migration algorithm. Finally, we conduct the implementation of a large-scale system. We investigate the influence of different system parameters on the performance of the migration algorithm including the time of periodical duration, slice size, the capacity ratio between two types of storage and available back-end bandwidth.

The structure of the chapter is as follows. Section 3.4 gives a description of a basic SVM migration algorithm and the other baseline algorithms. The preliminary comparison results and the issues of the basic SVM migration algorithm are provided in Section 2.3. Section 2.4 proposes an K-SVM migration algorithm. Section 2.5 shows the experimental result comparison between K-SVM and baseline algorithms. The results of a real large-scale implementation on a large cloud system are provided in Section 2.6 and related work is introduced in Section 3.7. Finally, the conclusion and future work are described in Section 3.8.

2.2 Basic SVM Migration Algorithm

In this section, we introduce a basic support vector machine (SVM) migration algorithm and also describe the basic steps of classification and migration of this algorithm. After that, some baseline algorithms are introduced as well. The terms and notations used in this chapter are defined in Table 2.1.

Table 2.1: Terms and notations used in this chapter

PT	performance tier (i.e., fast device)
CT	capacity tier (i.e., slow device)
C	the capacity of the whole system
Slice	the granularity of the unit for data migration
S_s	indicates the slice size (the default value is 200MB)
T	the time intervals to measure request density
Access density	the total number of IO accesses of one slice during the period T
N_s	total number of slices in the system ($N_s = C/S_s$)
N_{PT}, N_{CT}	numbers of slices in PT and CT, respectively
M_{PT}, M_{CT}	The sets of migration candidates. M_{PT}: the set of candidates of $PT \rightarrow CT$; M_{CT}: the set of $CT \rightarrow PT$
Training dataset ratio	training dataset ratio is calculated by the size of training dataset divided by N_s.
r_{PT}	the ratio between PT capacity and the total capacity. ($r_{PT} = N_{PT}/N_s$)
PT hit ratio	the number of requests in PT divided by the total number of requests.
BW	available back-end bandwidth and is indicated by the number of slices migrated in one period (# of slices/T)

2.2.1 Algorithm Description

SVM first proposed by Vapnik *et al.* [18] is a widely used supervised machine learning technique. SVM became popular because of its success in the handwritten digit recognition use case. SVM is a two-class classifier based on the two vectors from the training dataset. It can provide a hyperplane that maximizes the distance between two closest vectors in each of two classes [19]. For the hybrid storage system, the maximum distance between two clusters built by SVM can provide more precise classification/prediction and thus improve the performance and reduce the migration overhead.

In this work, we use SVM to categorize storage slices (slices are units of migration in hybrid storage systems) into two groups based on the historical workload access patterns. After classification, the slices will be migrated to a new location if its current location

Algorithm 1 Basic SVM Migration Algorithm: training

Input: C, S_s, T
Output: Hyperplane-Z

1: **procedure** TRAINING PROCEDURE
2: $N_s \leftarrow C/S_s$
3: Collecting access density of N_s slices in one T period
4: Sorting N_{PT} and N_{CT} slices based on the access density for PT and CT, respectively
5: Training dataset $(\mathbf{X}, \mathbf{Y}) \leftarrow$ top $x\% \times N_s/2$ slices in PT + the least active $x\% \times N_s/2$ non-zero slices based on the sorted access density. (default $x\% = 10\%$, so the size of training dataset is $x\% \times N_s$)
6: Training linear SVM based on training dataset $(\mathbf{X}, \mathbf{Y})$ to obtain a hyperplane-Z: $Z = AX + B$
7: **end procedure**

is mismatched with the SVM classification. The proposed SVM algorithm introduced in this section for storage migration is called a basic SVM migration algorithm (**basic-SVM**) in order to distinguish from the later introduced K-SVM migration algorithm (**K-SVM**).

There are two major steps in the basic SVM migration algorithm (training the basic SVM classifier and classifying and migrating). During each period T, the system records the access density (the number of times being accessed) of each slice. At the end of the period T, based on Algorithm 1 the training dataset is selected. The training dataset is used for building a new SVM classifier (new hyperplane). The classifier is used for classifying all slices based on the information collected last period. Finally, all migration candidates (storage slices) are determined. The migrating process happens in the next period $T + 1$.

Step I – Training: Algorithm 1 indicates the procedure of training. Assume a training dataset (X, Y) consisting of n points in the form of (X_1, Y_1) to (X_n, Y_n), where X_i is the i^{th} slice in the training dataset and Y_i is the label of the i^{th} slice and can be either 1 (Performance Tier (PT)) or -1 (Capacity Tier (CT)) indicating the class which the i^{th} slice belongs to.

For the training dataset, $x\%$ total slices are selected. For the basic SVM migration algorithm, $x\% = 10\%$ is set as the default value. For the later defined K-SVM algorithm, x will be adaptively changed. As indicated in Algorithm 1, the training dataset of the

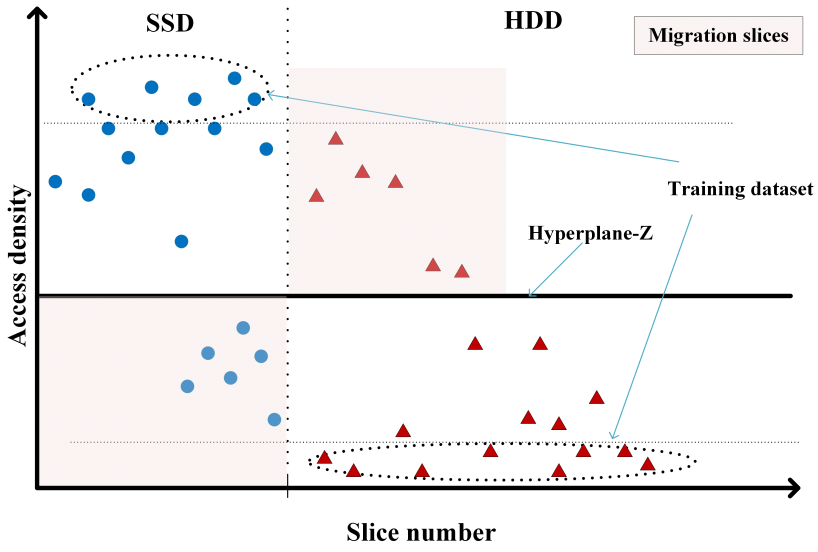

Figure 2.1: The basic SVM migration algorithm.

basic SVM is selected from the most active and the least active non-zero slices (slices with activities) of the performance and capacity tiers, respectively. In this way, it uses the most represented data to train the SVM.

After training, an SVM (Hyperplane-Z) is built, and then the Hyperplane-Z will be used for the classification process (testing). The output Hyperplane-Z in Algorithm 1 is a classifier that distinguishes which storage type the input slices should belong to as seen in Eq. 2.1.

$$\begin{cases} i^{th}\ slice \leftarrow PT, & \text{if Hyperplane-Z(i) == 1} \\ i^{th}\ slice \leftarrow CT, & \text{otherwise} \end{cases} \tag{2.1}$$

where Hyperplane-Z() function is obtained from Algorithm 1. i is the input slice number. PT means the performance tier. CT indicates the capacity tier. Therefore, if the output of Hyperplane-Z function with input i is 1, that means the i^{th} slice should be located in PT. Otherwise, the i^{th} slice should be located in CT.

Step II – Classifying and Migrating: After getting the Hyperplane-Z function from Algorithm 1, we start to identify the migration candidates and do the migration in the next period $T + 1$. Algorithm 2 indicates the procedures of classifying and migrating. First, based on the Hyperplane-Z function, for all slices

Algorithm 2 Basic SVM Migration Algorithm: classifying and migrating

```
1: procedure CLASSIFYING PROCEDURE
2:     while X_i ∈ PT slices do
3:         if Hyperplane-Z(i) ≠ Y_i then
4:             M_PT ← M_PT + X_i
5:             i ← i + 1
6:         end if
7:     end while
8:     while X_i ∈ CT slices do
9:         if Hyperplane-Z(i) ≠ Y_i then
10:            M_CT ← M_CT + X_i
11:            i ← i + 1
12:        end if
13:    end while
14: end procedure
15: end
16: procedure MIGRATING PROCEDURE
17:    # of migration slices = min(len(M_PT), len(M_CT))
18:    Ascending sorting M_PT
19:    Descending sorting M_CT
20:    for i ≤ # of migration slices do
21:        Exchange slices of M_PT(i) and M_CT(i)
22:        Updating the labels of slices of M_PT(i) and M_CT(i)
23:    end for
24: end procedure
25: end
```

in PT or CT, if the classification result of the i^{th} slice is not equal to its original label Y_i, then the i^{th} slice is added into its corresponding migration candidate set (M_{PT} or M_{CT}). During the migrating process, we first determine the number of migration slices by using the minimum number between the sizes of M_{PT} and M_{CT}. This is because some slices cannot be migrated/exchanged if the numbers of slices in M_{PT} and M_{CT} are not the same. By ascending sorting M_{PT} and descending sorting M_{CT} (Lines 13-17 in Algorithm 2), it promises that the most active slices in CT and the least active slices in PT are migrated first. The final step is to update the labels of migrated slices and those labels will be used for the next iteration period.

Figure 2.1 provides an example of the basic SVM migrating algorithm. According to Algorithm 1, the hyperplane-Z is trained based on the x% most and least active non-zero

Algorithm 3 Popularity-based Migration Algorithm

Input: T

Output: Migration candidates M_{PT}, M_{CT}

1: Collecting access density of N_s slices in one T period
2: Reversely sorting N_{PT} slices based on the access density for PT ($\mathbf{N}^i_{PT}$ indicates the i^{th} element in the sorted array)
3: Sorting N_{CT} slices based on the access density for CT ($\mathbf{N}^i_{CT}$ indicates the i^{th} element in the sorted array)
4: $i \leftarrow 0$
5: **while** $\mathbf{N}^i_{CT} > \mathbf{N}^i_{PT}$ **do**
6: $\quad M_{CT} \leftarrow M_{CT} + \mathbf{N}^i_{CT}$
7: $\quad M_{PT} \leftarrow M_{PT} + \mathbf{N}^i_{PT}$
8: $\quad i \leftarrow i + 1$
9: **end while**

slices in CT and PT, respectively. In Figure 2.1, after determining the hyperplane-Z, the migration candidates are classified as shown in shaded red regions. Finally, those candidate slices will be scheduled to be migrated to the region that they are supposed to reside.

According to the above description, the basic-SVM algorithm helps classify the migration slices which have similar or different features as the training dataset. The goals of the proposed migration algorithm are to improve the performance (higher PT region hit ratio (e.g., SSD hit ratio)) or to reduce total migration overhead (lower amount of migration data).

2.2.2 Baseline Algorithms

In this section, we introduce four baseline algorithms used for comparison in this chapter. One is the popularity-based algorithm, which is very popular with solutions from storage vendors. Some previous studies [20][21] can be simplified to the popularity algorithm. The basic idea is that the algorithm first collects the access density of each storage slice at each period T. Then, according to the access densities, the popularity-based algorithm is to exchange the slice of the highest access density in the CT region with the slice of the lowest access density in the PT region if the lowest value in the PT region is smaller than the highest value in CT region. The migration process will continue until the access densities of slices in the PT region are no longer smaller than the densities

of any slices in the CT region.

The HAT algorithm [22] is a migration algorithm considering both frequency and recency. The basic idea of HAT in a hybrid storage system (two types of disks) is that there is an LRU queue to record the recency of the historical data. The LRU queue size is the number of slices in the PT region (N_{PT}). The slice at its first-time access will be put into one LRU queue (LRU_Q). If the slice is reaccessed and located in LRU_Q, the slice will be put in PT_LRUQ. At the end of the algorithm, since the PT region only has the size of (N_{PT}), the first N_{PT} slices in PT_LRUQ should be put into the PT region. By comparing the locations of current slices, the migration slices will be put into M_{CT} and M_{PT}.

Another baseline algorithm is the Least Recently Used algorithm (LRU), which is a popular policy used in the eviction algorithm of memory cache. The LRU algorithm keeps the least recently accessed slices in an LRU queue for the PT region and keeps the most recently accessed slices in the MRU (most recently used) queue for the CT region. After period T, the algorithm exchanges the slices in the LRU queue with the slices in the MRU queue. The migration size of the LRU algorithm for each period is proportional to the sizes of MRU and LRU queues. By default, we set the LRU and MRU queue size to $N_s * 10\%$.

ChewAnalyzer algorithm [23] is another migration scheme for hybrid storage systems. The scheme is based on a hierarchical classifier [24] to classify the access patterns of workloads. They used different storage I/O workload characterization dimensions and the classifier analyze the access patterns step by step. To make a fair comparison, we simplify the ChewAnalyzer to a two-tier storage system. The first step is to classify the I/O density. Then, the second step is to distinguish the read and write performance. Finally, the sequence/randomness of workloads is classified. The high I/O intensive, write-intensive, and random workloads are assigned to PT (i.e., fast) devices and others are scheduled to CT (i.e., slow) devices. One of the baseline algorithms is popularity-based algorithm which is very popular with solutions from storage vendors. Some previous works [20][21] can be simplified to the popularity algorithm. The algorithm is defined in Algorithm 3. At period T, the popularity-based algorithm first collects the access density of each storage slice. Then, according to the access densities, the popularity-based algorithm is to exchange the slice of the highest access density in

CT region with the slice of the lowest access density in PT region if the lowest value in PT region is smaller than the highest value in CT region. The migration process will continue until the access densities of slices in PT region are no longer smaller than the densities of any slices in CT region.

2.2.3 Baseline algorithm II: HAT Algorithm

The HAT algorithm [22] is a migration algorithm considering both frequency and recency. The basic idea of HAT in hybrid storage system (two types of disks) is that there is an LRU queue to record the recency of the historical data. The LRU queue size is the number of slices in PT region (N_{PT}). As shown in Algorithm 4, the slice at its first time access will be put into one LRU queue (LRU_Q). If the slice is accessed again and it is also located in LRU_Q, the slice will be put in PT_LRUQ. It means the slice is labeled as a PT region candidate. At the end of the algorithm, since the PT region only has the size of (N_{PT}), the first N_{PT} slices in PT_LRUQ should be put into the PT region. With comparing the locations of current slices, the migration slices will be put into M_{CT} and M_{PT}.

2.2.4 Baseline algorithm III: LRU Algorithm

Another baseline algorithm is the Least Recently Used algorithm (LRU) which is a popular policy used in the eviction algorithm of memory cache. The LRU algorithm keeps the least recently accessed slices in a LRU queue for PT region and keeps the most recently accessed slices in the MRU (most recently used) queue for CT region. After period T, the algorithm exchanges the slices in LRU queue with the slices in MRU queue. The migration size of the LRU algorithm for each period is proportional to the sizes of MRU and LRU queues. By default, we set the LRU and MRU queue size to $N_s * 10\%$.

2.2.5 Baseline algorithm IV: ChewAnalyzer Algorithm

ChewAnalyzer algorithm [23] is another migration scheme for hybrid storage systems. The scheme is based on a hierarchical classifier [24] to classify the access patterns of workloads. They used different storage I/O workload characterization dimensions and

Algorithm 4 HAT migration Algorithm

Input: T
Output: Migration candidates M_{PT}, M_{CT}

for each request (Req_i) in T **do**
  Computing slice number (S_{Req}) of the request (Req_i)
  if S_{Req} in LRU_Q **then**
    $PT\_LRUQ \leftarrow PT\_LRUQ + S_{Req}$
  end if
  Put S_{Req} in LRU_Q
end for
for current slices ($slice_i$) in PT **do**
  if $slice_i$ is not at first N_{PT} of PT_LRUQ **then**
    $M_{PT} \leftarrow M_{PT} + slice_i$
  end if
end for
for current slices ($slice_i$) in CT **do**
  if $slice_i$ is at first N_{PT} of PT_LRUQ **then**
    $M_{CT} \leftarrow M_{CT} + slice_i$
  end if
end for

the classifier analyze the access patterns step by step. To make a fair comparison, we simplify the ChewAnalyzer to a two-tier storage system. The first step is to classify the I/O density. Then, the second step is to distinguish the read and write performance. Finally, the sequence/randomness of workloads is classified. The high I/O intensive, write-intensive, and random workloads are assigned to PT (SSD) devices and others are scheduled to CT (HDD) devices.

2.3 Performance of Basic-SVM Algorithm

2.3.1 Trace Characteristics and System Configuration

In the performance comparison, we use two types of traces, MSR Cambridge traces [25] and Systor'17 traces [26] to evaluate the performance of a hybrid system and migration overhead of all these algorithms. The trace characteristics are summarized in Table 2. Two metrics are used to indicate the performance of migration algorithms, PT hit ratio and total migration size. The PT hit ratio is defined as the number of requests

Table 2.2: Trace characteristics

	# of requests	Total request size (GB)	Trace length (h)	Maximum offset (GB)
MSR Cambridge traces [25]				
prn_1	1.04E+07	212.1	168	385.0
proj_1	1.47E+07	775.9	168	820.0
usr_1	3.63E+07	2135.4	168	820.0
usr_2	1.02E+07	441.8	168	530.0
src1_0	3.00E+07	1538.3	168	273.0
web_2	4.25E+06	263.6	168	169.0
stg_1	2.13E+06	85.5	168	101.7
mds_1	1.54E+06	88.7	168	474.0
proj_3	2.09E+06	20.9	168	220.0
Systor'17 traces [26]				
LUN0	6.38E+07	1607.8	36	4737.2
LUN1	6.27E+07	1794.9	36	4418.6
LUN3	6.54E+07	1638.6	36	4016.5
FIU traces [27]				
home3	9.18E+05	3.6	504	18.6
online	5.70E+06	21.7	720	7.9
webuser	7.73E+06	30.9	672	7.9
webmail	7.80E+06	29.7	720	18.2

satisfied by the slices in the PT region (i.e., fast device) divided by the total number of requests. The total migration size indicates how much data have been migrated between the two tiers. Therefore, a migration algorithm with a higher PT hit ratio and a smaller migration size will be better than others.

At the beginning of running traces, we preconditioned the storage system by writing all the slices that responded to the first portion of the requests to the PT region until the PT region is full. Then, the rest of the storage slices are written to the CT region (i.e., slow device). This precondition is practically used by industries to simply initialize a hybrid storage system. This preconditioning process is applied to all algorithms and is used in all simulations and experiments in this chapter.

2.3.2 Performance Comparisons

In this section, the performance comparisons between the basic-SVM algorithm, Popularity-based, HAT and LRU are made. In the experiments, the system capacity is set to 500GB, which contains 100GB SSD and 400GB HDD. The default slice size (S_s) is set

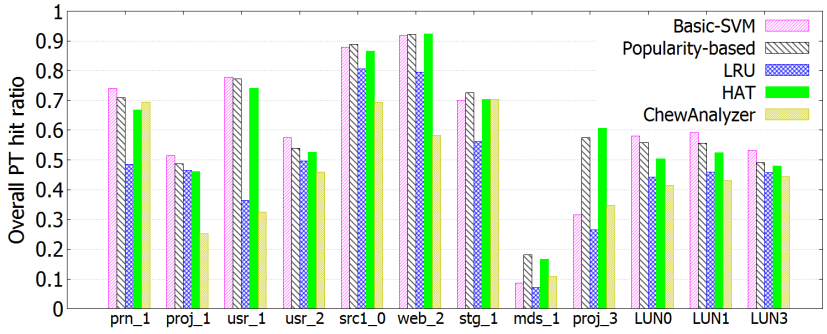

Figure 2.2: PT hit ratio comparison between basic-SVM algorithm, popularity-based, HAT and LRU algorithms.

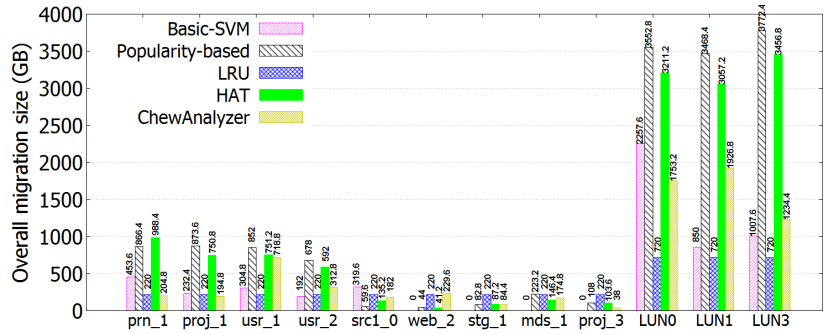

Figure 2.3: Migration size comparison between basic SVM, Popularity-based, HAT and LRU migration algorithms. "0" indicates there are no migration data.

to 200MB. Thus, there are a total of 2500 slices, 500 slices in SSD and 2000 in HDD. For those traces having larger maximum offsets than 500GB (like LUN0, LUN1 and LUN3), the offset is scaled into the range of 0-500GB, which is directly divided by a constant value. For example, for those traces from Systor'17, the offsets of traces are divided by 10. The configuration with scaling is equivalent to the configuration of 5TB total capacity and 2GB slice size without scaling. For the convenience of comparisons, the scaling is able to put the results of all traces in the same figures. For the basic-SVM algorithm, the training dataset is set to 10%. The size of the LRU queue is also set to 10%. The migration time interval (T) is 14 hours for MSR Cambridge traces and 1 hour for Systor'17 traces. By doing that, the total number of requests per T in each type of trace keeps similar.

As shown in Figures 2.2 and 2.3, the LRU algorithm has the worst overall PT hit

ratio. The reason is that the LRU algorithm always migrates the least recently used slices and cannot reflect the characteristics of workloads. Therefore, it causes a much low PT hit ratio. The migration only happens for each period. So, the LRU policy is capable of improving the cache hit ratio by immediately replacing the most recent accessed data but is not good at in the storage migration scheme. For the other four algorithms, the overall PT hit ratio for most of the traces is similar, while the basic SVM algorithm achieves a much smaller overall migration size. This is because ChewAnalyzer, HAT and Popularity based schemes use the constant schemes to determine the access patterns. Therefore, they cannot dynamically follow the change of workloads and they achieve either a lower PT ratio or higher migration overhead than the basic SVM scheme. However, there are three exceptions. For the traces mds_1 and proj_3, the basic SVM migration algorithm only gets about 8% and 31% overall PT hit ratio, respectively. They are much smaller than the PT hit ratios of the popularity-based and HAT algorithm (18% and 17% for mds_1, and 57% and 60% for proj_3). For trace src1_0, although the basic-SVM, popularity-based and HAT algorithms achieve similar PT hit ratio, the basic-SVM needs to transfer 5x and 3x larger migration size than the popularity-based and HAT algorithms. According to these three exceptions, the issues of the basic SVM migration algorithm are investigated and discussed in the following subsection. After that, a new K-SVM migration algorithm is proposed for solving those issues in Section 2.4.

2.3.3 Issues of Basic-SVM Migration Algorithm

After investigating the three traces that the basic-SVM algorithm has worse performance than that of popularity-based algorithm, we found that the issue is selecting improper training datasets.

SVM Mis-Classification

In a hybrid storage system, we always want to migrate the frequently accessed slices to SSD and migrate the least accessed slices to HDD. At the beginning of applications, writing data first to SSD makes sure that the SSD contains most of highly accessed slices. However, running for a while, according to the changes of trace access patterns, it is possible that HDD contains many highly accessed slices (only considering slices

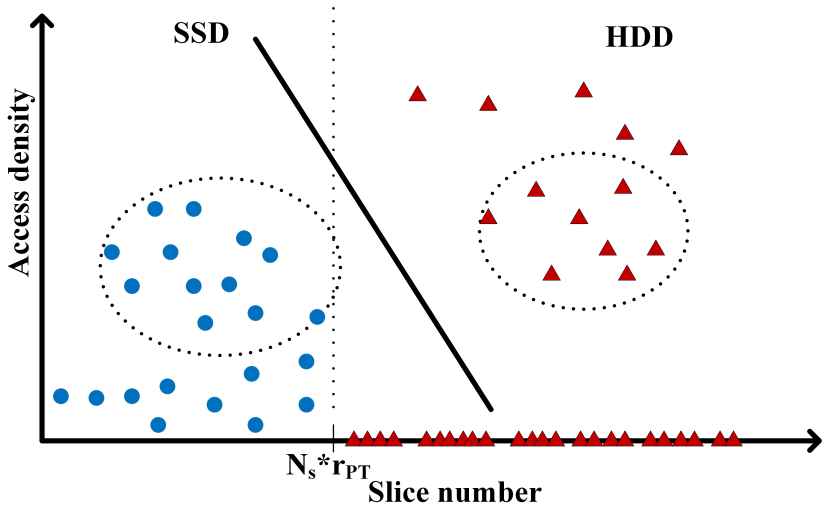

Figure 2.4: The scenario that the basic-SVM mis-classifies slices for SSD and HDD.

with activities). Thus, the hyperplane generated by the basic-SVM has a negative slope as seen in Figure 2.4. In this case, the hyperplane tries to migrate highly accessed slices to HDD and migrate lower accessed slices to SSD due to the negative slope. Consequentially, highly accessed slices are kept in HDD. Finally, this scenario results in the low PT hit ratios as in the traces mds_1 and proj_3.

Improper Training Dataset

As discussed in Section 2.3.2, the basic-SVM migration algorithm ended up with a larger migration size for trace src1_0. The migration size is determined by the SVM hyperplane which is trained by a selected training dataset. Thus, we first investigate the relationship between the overall migration size and training dataset ratio under the system as configured and discussed in Section 2.3.2.

As shown in Figure 2.5, the overall trend of PT hit ratio keeps roughly flat with the increasing training dataset ratio for all traces. However, in Figure 2.6 the overall migration sizes are changed tremendously and irregularly for different traces with increasing training dataset ratio. The maximum migration size can reach more than 10X than the minimum migration size in Figure 2.6. Therefore, the migration size is highly related to the training dataset ratio. Based on Figure 2.6 the curves are so irregular and it seems

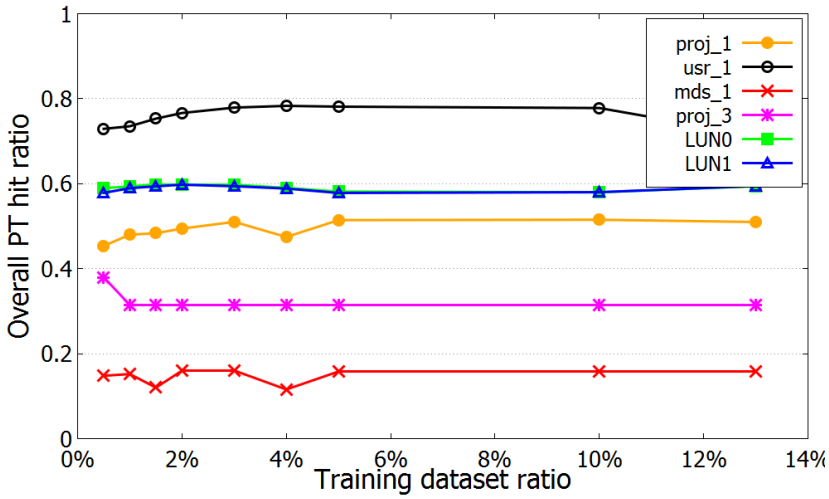

Figure 2.5: Relationship between the overall PT hit ratio and the training dataset ratio.

hard to find a rule for picking up a proper training dataset ratio for a specific trace.

Moreover, to find the relationship between migration size and PT hit ratio, we vary the training dataset ratio to obtain different migration sizes for the basic-SVM migration algorithm. As shown in Figure 2.7, the PT hit ratio is increased with the raising migration size at the beginning and then the overall PT hit ratios become saturated. The goal of migrating data in a hybrid system is to achieve a higher PT hit ratio while maintaining a small migration size. So, those so-called **balanced points** in Figure 2.7 have good trade-offs between the migration size and the PT hit ratio. As for the issue of large migration sizes for the basic SVM migration algorithm in Section 2.3.2, it is because the result of the basic-SVM algorithm locates far away from the balanced point of src1_0 (at the right side) in Figure 2.7. The reason for having a large migration size is caused by an improper training dataset due to the constant training dataset ratio. For different traces, the request access patterns are different and the same training dataset ratio is not a good choice. Moreover, even for the same trace, the request access patterns are changed and different at different iterations. Therefore, the training dataset ratio directly affects the performance of a migration algorithm (the PT hit ratio and total migration size). A proper training dataset ratio is useful for solving the issue of large migration sizes (investigated in Section 2.5).

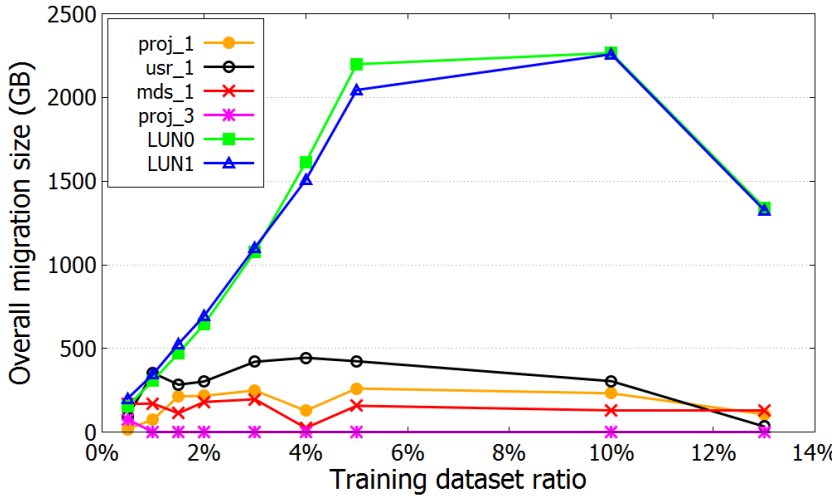

Figure 2.6: Relationship between the overall migration size and the training dataset ratio.

2.4 K-SVM Migration Algorithm

In this section, a modified SVM migration algorithm called K-SVM migration algorithm (**K-SVM**) is introduced to remedy the two issues of the basic SVM algorithm as discussed in Section 2.3.3.

The proposed K-SVM migration algorithm is shown in Algorithm 5. Compared to the basic-SVM algorithm in Algorithm 1, the main differences are the training dataset selecting (Lines 4-8 in Algorithm 5). The proposed K-SVM has two advantages compared to other schemes. One is to accurately predict the workloads and then can achieve a better hit ratio. Moreover, the precise classification can help reduce the migration overhead since those incoming requests are put in the 'right' location based on the classification and there is no need to migrate those slices. As a result, the migration cost will be reduced. In this chapter, the maximum migration is limited by the available bandwidth. In the following results, we assume that the system has adequate bandwidth to migrate slices between two tiers.

To remedy the improper training dataset issue, the basic idea is to include the most representative slices as many as possible into the training dataset for SVM. For example, we want to include most of the relatively highly accessed slices in the training dataset of the PT region. By doing that, those relatively high accessed slices can effectively

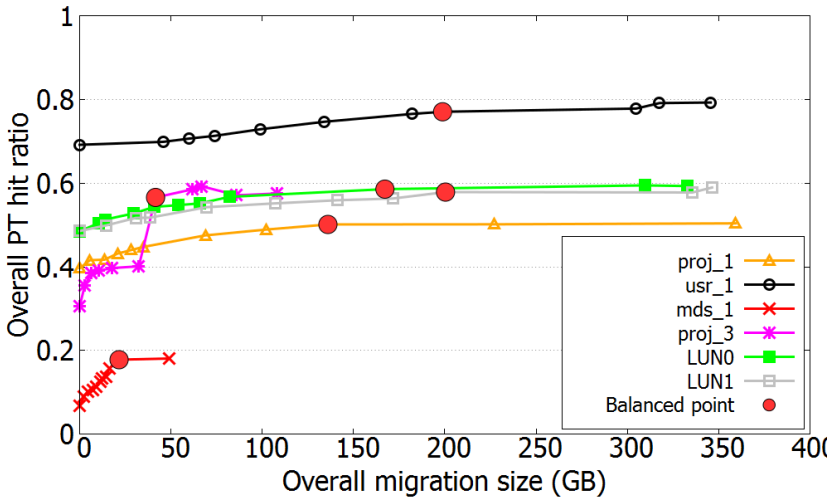

Figure 2.7: Relationship between overall migration size and overall PT hit ratio for the SVM migration algorithm.

represent the feature of the PT region. Additionally, those slices in the training dataset will not be migrated due to the feature of SVM and thus it potentially reduces the migration size. Similarly, the training datasets should also exclude the slices which cannot represent the feature of the region. Therefore, by replacing a constant training dataset ratio, we use the K-Means clustering algorithm [28] to group the similar slices in PT and CT regions, respectively (K=2 used in this chapter). The K-Means clustering algorithm is used for PT and CT regions, respectively, with one dimension input (access density).

In some cases, one or two slices located in the PT region have really high access frequencies than others. However, we do not want to only use one or two points to represent the PT region. Therefore, to overcome those outliers, we force the top cluster containing at least 0.2% slices as shown in Lines 4-7 in Algorithm 5. Therefore, by using the modified K-Means clustering algorithm, the training dataset is adaptively selected by the algorithm itself. As a result, compared to other algorithms the K-SVM algorithm achieves smaller migration sizes and higher PT hit ratios in Section 2.5.

Algorithm 5 K-SVM Migration Algorithm: training

Input: C, S_s, T
Output: Hyperplane-Z

procedure TRAINING PROCEDURE
    $N_s \leftarrow C/S_s$
    Collecting access density of N_s slices in one T period
    Sorting all slices in PT
    Remove top 0.2% slices
    Do K-Means clustering for PT region (K=2).
    Adding the removed top 0.2% slices to the cluster at the top position.
    Do K-Means clustering for CT region (K=2).
    Training dataset $(\mathbf{X}, \mathbf{Y}) \leftarrow$ all slices at the top cluster of PT + all slices at the bottom cluster of CT
    Training linear SVM based on training dataset $(\mathbf{X}, \mathbf{Y})$ to obtain a hyperplane-Z: $Z = B$
end procedure

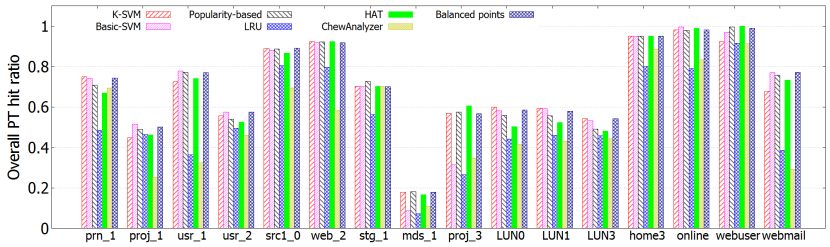

Figure 2.8: The PT hit ratio comparisons between K-SVM and other algorithms.

2.5 Experimental Result

To find how well the K-SVM algorithm is applied to the data migration problem of hybrid storage systems, we compare the performance of the K-SVM algorithm with that of basic SVM, popularity-based, HAT and the performance of balanced points (discussed in Section 2.3.2). A new type of traces (FIU trace [27] as shown in Table 2.2) is added in this experiment. The system configurations are set to the same as the configuration in Section 2.3.2.

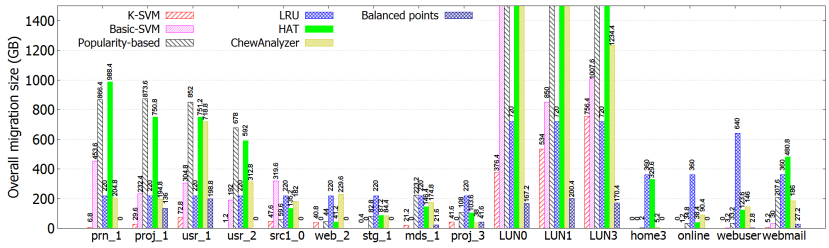

Figure 2.9: The migration size comparisons between K-SVM and other algorithms.

2.5.1 Overall Performance Comparison

The performance and overhead comparisons are shown in Figure 2.8 and Figure 2.9 respectively. Among all algorithms, the LRU algorithm has the worst overall PT hit ratio. This is because the LRU algorithm always migrates the least recently used slices and cannot reflect the characteristics of workloads. Therefore, it causes a much low PT hit ratio. The migration only happens for each period. So, the LRU policy is capable of improving the cache hit ratio by immediately replacing the most recent accessed data but is not good at in the storage migration scheme. In the future experiment comparisons, we do not compare the LRU algorithm by varying system parameters. The K-SVM achieves similar or a little lower PT hit ratios as the balanced points. For the migration size, the balanced point results always have the lowest values among most traces. For traces proj_1 and usr_1, the balanced point has larger migration sizes but higher PT hit ratios than the proposed K-SVM algorithm. For the rest of the traces, the newly proposed K-SVM algorithm achieves close migration sizes to the balanced points and obtains much smaller migration sizes compared to the basic-SVM, popularity-based, LRU, HAT and ChewAnalyzer algorithms.

In summary, although for some traces, the K-SVM algorithm has slightly larger migration sizes compared with the results of balanced points, it remedies the issues described in Section 2.3.3. Moreover, reducing the migration size is significant for all traces (2x - 8x on average). Therefore, the K-SVM migration algorithm effectively selects a proper training dataset for the SVM classifier and gains very close solutions to the balanced points which have the smallest migration size and the highest PT hit ratio.

2.5.2 K-SVM Overhead Discussion

The overhead of the K-SVM scheme mainly comes from two aspects. One is the metadata overhead of recording collected trace information. The second one is the computation overhead of machine learning algorithms. Assume the total capacity of PT and CT tiers are 500 GB (PT: 100GB and CT: 400GB). The slice size is 200MB. The metadata information only has about 16KB. Compared to the total 500GB capacity, the metadata overhead will have little influence on the systems. For the other overhead, we investigate the execution time of the training process. As seen in Table 2.3, the training time is varied from 3.06ms to 211.79ms as varying the slice size. Compared to the period T (hours), the training time of the K-SVM scheme is acceptable.

Table 2.3: Training time of the K-SVM scheme with varying slice size

slice size (MB)	50	100	200	500	1000
Training time (ms)	211.79	56.76	16.98	5.45	3.06

2.5.3 Comparison with Classic Caching Policies

We compared several caching algorithms (ARC [29], LeCaR [30], LRU and LFU) to the proposed SVM scheme described in Section 3.4. As indicated in Table 2.4, the proposed scheme achieves much better hit ratios for most of the traces. The reason is that the hybrid system schemes monitor trace information on both PT and CT and thus can achieve better performance than caching algorithms. Moreover, the write-back operation (element eviction to slow devices) is not considered for the caching algorithms, which makes the performance of caching algorithms even worse. To make fair comparisons, in the following sections, we mainly focus on the algorithms for hybrid storage systems.

2.5.4 Effect of Space Capacity Ratio between PT and CT

In this subsection, we investigate the effect of different device capacity ratios. We keep the configuration the same as the previous subsection with the slice size as 200MB.

As seen in Figures 2.14 and 2.19, the PT hit ratios are increased with the increased PT capacity (SSD capacity). The K-SVM, popularity-based, HAT and ChewAnalyzer

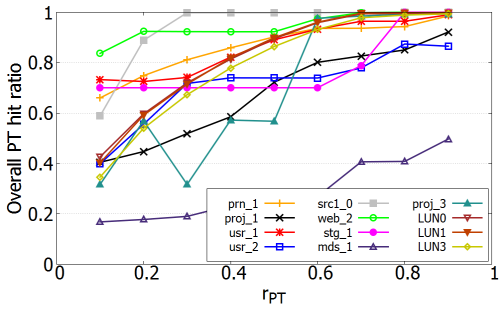

Figure 2.10: K-SVM

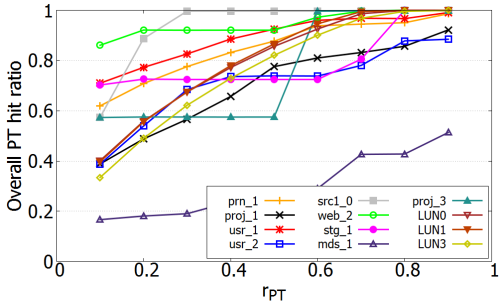

Figure 2.11: Popularity-based

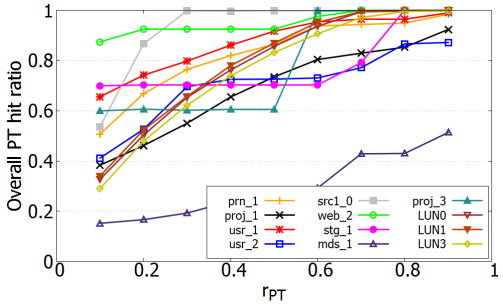

Figure 2.12: HAT

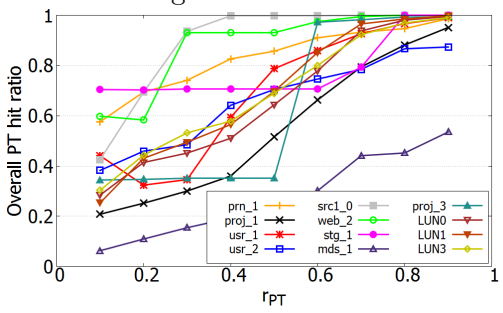

Figure 2.13: ChewAnalyzer

Figure 2.14: The PT hit ratio with varying SSD (PT region) capacity.

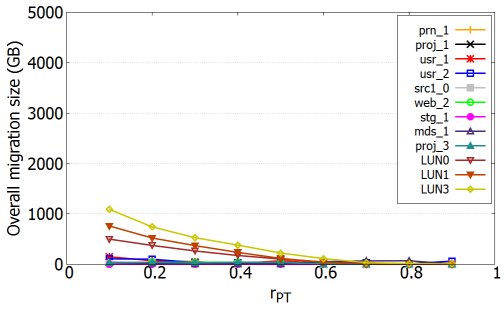

Figure 2.15: K-SVM

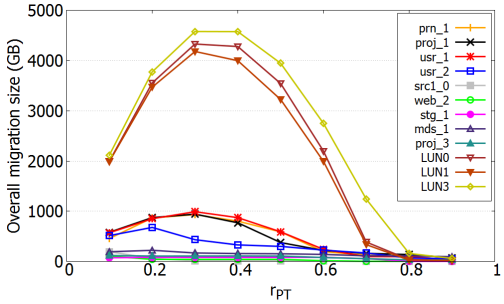

Figure 2.16: Popularity-based

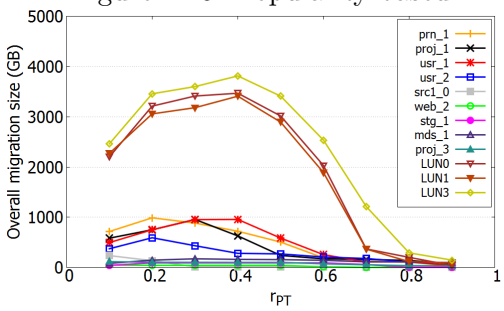

Figure 2.17: HAT

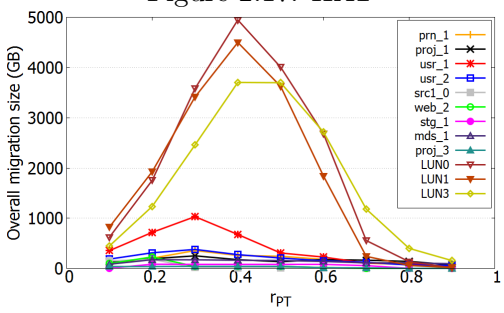

Figure 2.18: ChewAnalyzer

Figure 2.19: The migration size with varying SSD (PT region) capacity.

Table 2.4: Hit ratio comparisons between caching algorithms and the proposed scheme with 100GB PT capacity

	ARC	LeCaR	LRU	LFU	K-SVM
prn_1	61.86%	61.86%	61.86%	61.86%	74.87%
proj_1	9.22%	8.00%	8.00%	9.22%	44.74%
usr_1	66.60%	64.76%	64.76%	66.60%	72.53%
usr_2	12.73%	11.07%	10.89%	12.73%	55.59%
src1_0	52.17%	49.91%	49.91%	61.67%	88.88%
web_2	74.97%	74.97%	74.98%	74.98%	92.41%
stg_1	6.80%	6.80%	6.80%	6.80%	70.10%
mds_1	5.05%	5.05%	5.05%	5.05%	17.81%
proj_3	72.07%	72.07%	72.07%	72.07%	56.94%

algorithms achieve similar PT hit ratios. For the migration size, our proposed K-SVM migration algorithm achieves about on average 32X smaller migration size when compared to the popularity-based, HAT and ChewAnalyzer algorithms.

Moreover, we investigate the influence of the PT capacity ratio on the migration size for each algorithm. For the popularity-based, HAT and ChewAnalyzer algorithms, they achieve a similar trend of migration size, which is that the migration size first goes up and reaches the peak values when r_{PT} is about 0.35 for Popularity based and about 0.4 for HAT and ChewAnalyzer. Then, the migration size decreases with the increased SSD (PT region) capacity. This is because at the around middle points, the numbers of migration candidates for PT and CT slices become similar and thus the migration size reaches the peak values. Before or after the middle points, either the number of PT candidates or the number of CT candidates is reduced. The mismatched number of candidates results in a smaller migration size. With increased PT capacity, the K-SVM algorithm migrates less amount of data. This is because a larger PT space can store more data and thus the number of migration candidates becomes less. The K-SVM algorithm efficiently selects the migration candidates and keeps the migration size small while maintaining similar PT hit ratios as other algorithms.

Table 2.5: List of applications used on the single hybrid storage system test bed

Tenant	Application
A	Oracle, SAP, VMware
B	Home Directory
C	High Performance Computing
D	Virtual Desktop (VDI), Hyper V
E	SharePoint, Web Farm

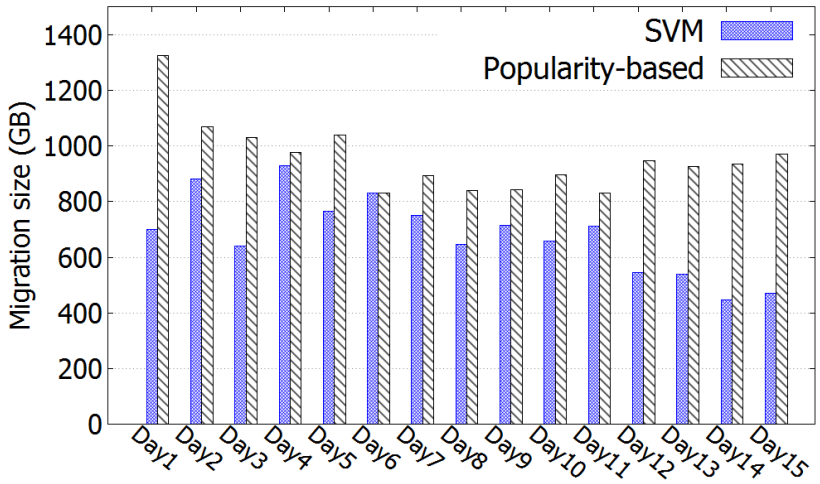

Figure 2.20: Migration size comparison between SVM and popularity-based algorithms

2.6 Large-scale System Implementation

The prototype of our proposed SVM and popularity-based algorithms are applied to a real enterprise hybrid system. We compared popularity-based and SVM algorithms since based on all previous discussions the popularity-based algorithm performs much similar to the HAT algorithm. The experiments were conducted first on the traces gathered from a live system with multi-tenant 26 different applications running at an enterprise lab system for 31 days. The application list is shown in Table 2.5. The total storage capacity in the test environment for Hybrid Storage System used was 1 PB with 100TB in SSD and 900TB in HDD.

The tests were performed with 47% storage consumed and at the end of the 15 days period and the total capacity used by the system was 49%. Four metrics are considered 1) the average latency as experienced by the host before migration (**pre-migration**); 2)

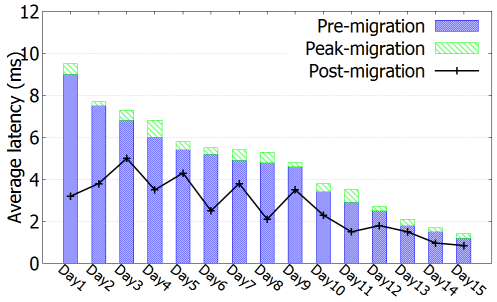

Figure 2.21: SVM

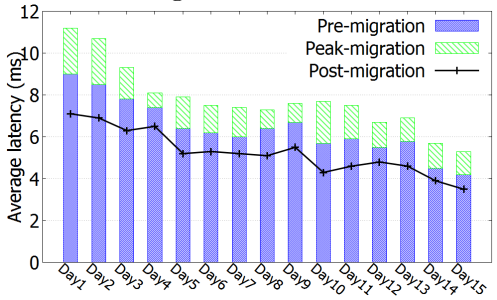

Figure 2.22: Popularity-based

Figure 2.23: Latency impact (prior, during and post migration)

latency spikes when data is actually migrated (**peak-migration**); 3) the average latency as experienced by the hosts after migration (**post-migration**); and 4) **Migration size**.

As seen in Figure 2.20, the system with the popularity-based algorithm moves about 0.95TB data per day and the system with the SVM algorithm only moves about 0.68TB data per day on average. Thus, the SVM algorithm achieves about 40% data migration reduction compared to the popularity-based algorithm. For the average latency in Figure 2.23, both algorithms are able to reduce the latency of the overall system after migrating. On average, the latency of the system with the SVM algorithm drops from 4.5ms to 2.7ms. While the system with the SVM algorithm only achieves the drop from 6.4ms to 5.25ms. The SVM algorithm reduces the average latency by about 70% compared to the popularity-based algorithm. Moreover, the transient spikes in the popularity-based algorithm are as high as 27% on average, which is much larger than the spikes in the SVM algorithm (only 9.1%).

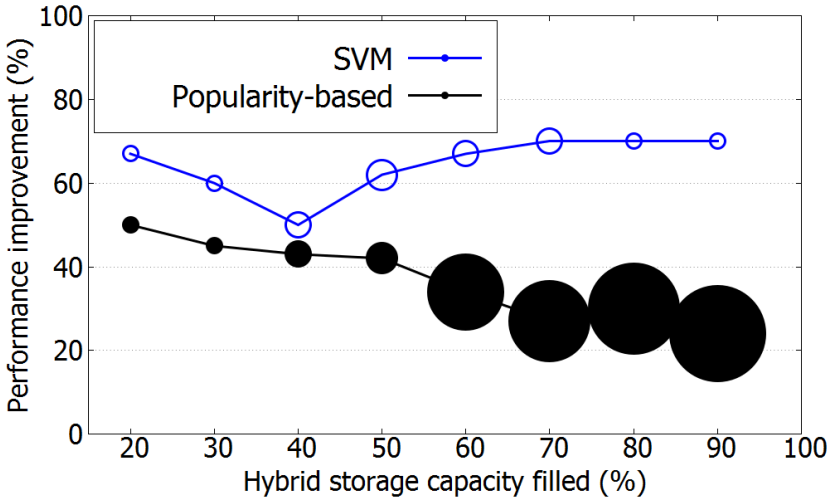

Figure 2.24: Performance improvement and migration sizes with various storage capacities for SVM and popularity-based. (the sizes of bubbles indicate the migration size)

Additionally, we examined the impact of the storage capacity on both popularity-based and SVM algorithms. As shown in Figure 2.24, the SVM algorithm achieves a higher performance improvement at all different available capacities than the popularity-based algorithm. Meanwhile, the SVM achieves less migration size than the popularity-based algorithm as well.

2.7 Related Work

A hybrid storage system combining slow and fast devices have the advantages of cost-effectiveness, higher performance and longer endurance. Thus, it has gaining popularity and attracting research interests since the beginning of this decade [31, 32, 33]. With the advent of flash memory-based SSDs now and Non-Volatile Memory (NVM) in the near future, research on caching and tiering algorithms to improve and deliver Quality of Service (QoS) of storage systems has been extensively conducted as well [34, 35, 36, 37, 38, 39].

The data tiering management normally can be classified into two categories: file-level and block-level. For the file-level migration, the storage manager has the information

about application files and thus it can precisely migrate data based on the characteristics of applications [40, 41, 11]. However, compared to the block-level migration, the file-level migration is less efficient and has a larger migration overhead due to its migration granularity and the migration decision to be done by file manager. For block-level migration, Guerra *et al.* [13] proposed the Extent-based Dynamic Tiering (EDT) tool that contains two components: a configuration adviser (EDT-CA) and a dynamic tier management (EDT-DTM). The EDT-CA determines the extent placement based on a fixed utility function. The EDT-DTM manages the extent placement and migration via monitoring active workloads. ExaPlan [12] achieves a low mean response time by using a queuing model. HybridStore [32] provides a cost-efficient storage configuration for specific workloads and is able to reduce the response time for the random-write dominant workloads. These studies have two major differences from our work. One is that they used simple heuristic methods to solve the migration problem. The other one is that they focus on three or more types of devices and also consider the prices of devices. In this chapter, we exploit the possibility of using machine learning approaches.

2.8 Conclusion

By applying known machine learning approaches to the storage domain, an entire new set of tools can be applied to solve data tiering problems. In this chapter, we propose a migration algorithm based on Support Vector Machine (SVM) and demonstrate the effectiveness of this algorithm to solve an optimization problem in the enterprise storage domain. Moreover, the proposed K-SVM migration algorithm involves K-Means clustering to dynamically select a proper training dataset. The proposed algorithm can tremendously reduce the size of the migration data. Finally, the results of a real implementation indicate that the ML-based algorithm reduces the volume of migration data by about 40% and achieves 70% lower latency compared to other algorithms.

Chapter 3

E-VM: An Elastic Virtual Machine Scheduling Algorithm to Minimize the Total Cost of Ownership in a Hybrid Cloud

3.1 Introduction

A hybrid cloud that combines both public and private clouds has the advantages of improved security, scalability, and guaranteed SLA (Service-Level Agreement) at a lower cost than a separate private or public cloud. The desired outcome of a hybrid cloud is to ensure the SLAs of all virtual machines (VMs) running on either private or public cloud at any given time. To accomplish this goal, VM migrations from private to public cloud are required when the private cloud's utilization nearly reaches its limits. Moreover, VM migrations from the public to the private cloud should also be considered to reduce the cost when the private cloud's resources are underutilized. The existing studies rarely consider VM migrations in a hybrid cloud environment with dynamically changed VM workloads. From an enterprise's perspective, these migrations are necessary to minimize the cost of utilizing public clouds and guarantee SLAs of VMs in a hybrid cloud environment.

In this chapter, we propose an elastic VM allocation and migration algorithm for a hybrid cloud, called E-VM, to fully utilize the resources in a private cloud and to minimize the cost of using a public cloud while guaranteeing the SLAs of all VMs. The E-VM considers the bi-direction migration between private and public clouds. Two components, VM-predictor and VM-selector, are designed and implemented in E-VM to determine if a migration has to be triggered between private and public clouds and which VMs will be migrated to the opposite cloud, respectively. Moreover, E-VM is designed based on the existing public cloud pricing models and can be easily adapted to any cloud service provider. According to simulator results based on a set of captured industrial VM traces/workloads and additional experiments directly on a real-world hybrid cloud, the proposed E-VM can significantly reduce the total cost of using the public cloud compared to the existing VM migration schemes.

Cloud computing has become a prevalent platform for offering computing services and storing data for various applications. One of the vital cloud computing features is its scalability and elasticity of both computing and storage resources. Many startups and small companies have used public clouds as their IT infrastructure. Recently more and more enterprise companies have begun to use public clouds and their private cloud to meet their computing and storage demands. For example, cloud tenants (e.g., IBM, Hewlett Packard Enterprise (HPE) and Dropbox) purchase cloud computing services from cloud providers (e.g., Amazon Web Service (AWS) [1], Google Cloud [4], and Microsoft Azure [3]) to enhance their private cloud capabilities. Pay-As-Your-Grow (PAYG) basis offers cloud tenants benefits of saving the cost compared to the traditional software license plus maintenance contract. The public cloud of a hybrid cloud can be used to offer additional computing and storage resources if needed.

The transformative networked cloud computing can be categorized into private cloud, public cloud, and hybrid cloud. The private cloud can base on the procured best of breed compute, network, and storage from different vendors. The public cloud is the cloud computing model that offers services via the Internet. Cloud providers are responsible for resources management, software deployment, and server/hardware maintenance. The tenants of the public cloud can put their applications in the cloud by paying for the usage of corresponding resources. However, these two types of clouds

have their own advantages and disadvantages. For example, a private cloud lacks scalability and elasticity when dealing with bursty demands from workloads. The public cloud is simple to use and flexible to scale up, but its cost is higher than the private cloud in a longer duration.

A hybrid cloud, a mixture of private and public clouds, can offer both advantages of private and public clouds. First, a hybrid cloud can improve security and privacy by putting sensitive data in the private cloud. Meanwhile, the public cloud component offers elasticity and scalability to meet the service level agreements (SLAs) of users/VMs when the demands of workloads grow up, or a demand spike happens. However, currently many issues of hybrid cloud are not thoroughly investigated. For example, how to migrate virtual machines between private and public clouds? What is a cost-efficient way to do VM migration? And, what is the optimal public cloud pricing strategy for the concerned workloads?

In this chapter, we focus on minimizing the total cost of ownership (TCO) of an enterprise company that uses a hybrid cloud. To simplify the problem, we assume that each VM may contain multiple containers for one independent service. Those enterprises normally own a private cloud with limited resources and also run VM in the public cloud when having the increased or unpredictable bursty demands from VMs such as utilizations of CPU, memory, storage, etc. Thus, some VMs must be dynamically migrated to the public cloud to guarantee the specified SLAs of all services. The public cloud will charge based on the usages of resources by the VMs and obviously the payment for the public cloud may significantly increase the total cost of ownership (TCO). How to minimize the TCO in a hybrid cloud becomes a crucial issue and interesting research topic. To address this problem, three main questions need to be answered: 1) When to trigger a VM migration? 2) Which direction the VM migration should be: from private cloud to public cloud or from public cloud to private cloud? And 3) Which VMs should be selected for the migration?

To answer all three questions, we propose a scheduling algorithm for hybrid cloud, called E-VM, to minimize the TCO when using a hybrid cloud. The E-VM consists of two major components: VM-predictor and VM-selector. VM-predictor determines when to trigger a migration between private and public clouds and which direction for migration. VM-selector decides which VMs are selected to be migrated. Moreover, the

VM-selector takes the public cloud's pricing model into consideration so that the E-VM can optimize the VM migration to achieve a much lower TCO than others.

The organization of the chapter is as follows. Section 3.2 introduces the background of different types of cloud environments and the pricing models of modern public clouds. Section 3.3 formulates the issue that we try to solve and introduces the motivation of this work. The proposed scheme is described in Section 3.4. The conducted experiments are discussed in Section 3.5 and the experimental results in a real system are shown in Section 3.6. Section 3.7 briefly summarizes the related work. Finally, a conclusion is drawn in Section 3.8.

3.2 Background

3.2.1 Public, Private and Hybrid Cloud

Public Cloud: The "pay as you use" consumption model for the public cloud is very attractive to software developers to get a fast start. At the initial stage of development, the required CPU, memory, and network bandwidth as well as the test data are relatively small. So, convenience over cost is a reasonable and favorable trade-off. In this chapter, we assume the applications or services developed are based on VMs. Different pricing strategies/models may affect the virtual machine placement strategies. In general, public cloud is a good choice for a) bursting temporary workloads, b) test and development environments which require a fixed set of data to be exported/imported during test and development processes, and c) low to modest amount of storage usage.

Private Cloud: Prior to the public cloud's popularity, enterprises procured best of breed compute, storage and networking from different vendors, with their proprietary set of tools to support applications and manage/integrate those disparate components together. Therefore, it is much more complicated when compared with the experiences of the public cloud. Also, without the public cloud's support, the private cloud has to be designed with the consideration of the required peak demand for resources and thus can be very costly. However, without a private cloud, the enterprises risk losing their own internal software developers and the controllability of their data.

Since the equipment is already physically procured in private cloud, the focus of the private cloud is to maximize the utilizations of these resources. However, if some

resources in private cloud are saturated, the SLAs of certain VMs may be violated. Therefore, it is important to identify workloads that are best migrated to the public cloud to alleviate the saturation of resources while keeping the total cost down.

Hybrid cloud: Essentially, the public cloud provides a) simplicity and b) cost advantage at a smaller scale. The private cloud provides a) cost at scale, b) close control of data and resources, and c) possible performance advantage. The most remarkable technology which makes cloud computing feasible is the advent of server virtualization. With server virtualization (i.e., VMs), it is now possible to move VMs between servers in a datacenter or from the private cloud back and forth to the public cloud. Therefore, we can combine the public cloud's benefits with that of the private cloud to provide a) highly scalable, b) cost-effective, and c) high-performance cloud environment with guaranteed SLAs. In a hybrid cloud, either for cost, performance or control reasons, the need to move VMs between public and private clouds also requires data to be migrated across extremely distributed domains.

3.2.2 Pricing Schemes in Cloud Computing

Currently, many public cloud services like Amazon Web Service (AWS) [1], Google Cloud [4], and Microsoft Azure [3] provide several types of cloud computing services including compute, database, analytic, blockchain, etc. The instance types cover from various combinations of CPU, memory, storage, and networking capacities to satisfy requirements from different tenants. By reviewing the pricing models from different cloud providers, we find them sharing similar services and the pricing models can be categorized with two perspectives. One is based on the types of instances, which provide different resource configurations/combinations. For example, general-purpose instances contain balanced CPU, memory, storage, and network resources. The type of memory-

Table 3.1: Pricing model in Amazon Web Services (AWS) [1]

	vCPU	Memory (GiB)	Cost (per hour)
m5.large	2	8	$0.096
m5.xlarge	4	16	$0.192
m5.2xlarge	8	32	$0.384
m5.4xlarge	16	64	$0.768
... ...			
General Purpose SSD (gp2)			$0.1 per GB-month

or CPU- optimized instances includes more memory or CPU resource than other types. Therefore, they can be used for supporting different applications.

According to the property of "pay as you use" in the public cloud, instances with different pricing plans have various pricing rates. Table 3.1 shows the pricing rates of the general-purpose instances with the on-demand plan. Moreover, the users are also charged by transferring their data out (not transferring data in) from the cloud as shown in Table 3.2. Different locations (e.g., Ohio or California) also have different pricing rates. Some cloud providers give a separate storage price rate called Elastic Block Store (EBS) for different types of storage devices associated with an instance as seen in Table 3.1. Our study in this chapter is based on a pricing model shown in Table 3.1. We assume that the cloud providers give a pool of resources, and a resource in the pool can be discretely and incrementally added. Moreover, the scheme is independent to the pricing model and can be easily applied to other types of price models.

Finally, the SLAs of all VMs are hard to be fully satisfied due to the possible failures of hardware devices [42]. Therefore, cloud providers always provide an advertised annual or monthly failure rate like 0.00001 percent, which means the total downtime in one month or one year is about 0.00001% (equivalent to about five minutes of downtime per year). If the downtime is higher than the advertised value [2], there will be a penalty for the cloud providers (discounts for users) as seen in Table 3.3. In this chapter, we follow a similar penalty model as Table 3.3. Since the private cloud has limited resources, it is also possible that some misclassification happens and the management does not migrate some VMs to the public cloud on time. As a result, the utilization of VMs in the private cloud is overflowed (i.e., resources are saturated). So, there will be a penalty when the violations of the SLAs happen. The percentage of downtime is formulated by the total number times of detected resource overflows divided by the total number of times the

Table 3.2: Pricing of transferring VMs out of clouds in AWS

Amazon [1]		Microsoft [3]		Google [4]	
Size	$/Month	Size	$/Month	Size	$/Month
<1GB	0	<5GB	0	<1TB	0.12
<9.99TB	0.09	<9.99TB	0.087	<10TB	0.11
next 40TB	0.085	next 40TB	0.083	10TB+	0.08
next 100TB	0.07	next 100TB	0.07		
>150TB	0.05	next 350TB	0.05		

monitoring and evaluating were done and the penalty cost is proportional to the daily cost of private cloud resource usage.

Table 3.3: SLA violation penalty model [2]

Monthly Uptime Percentage	Penalty (service credit)
>99.9%	0
>99%	10%
>95%	25%
<95%	100%

3.3 Problem Description

In this section, we describe the problem that we intend to solve. We also present two baseline approaches that will be compared with our approach later. The problem is based on VM management in a hybrid cloud from an enterprise perspective. The enterprise maintains a hybrid cloud including its own private cloud and an outside public cloud. The private cloud has a fixed amount of resources, and the public cloud can always provide adequate resources as the enterprise asked and plans to pay for. A set of long-running virtual machines (VMs) are considered in the hybrid cloud. These VMs support different types of user applications and services. The utilizations of each VM in terms of CPU, memory and storage are dynamically changed since the required resources of different tasks are up and down during different hours and days. The enterprise manages these VMs to satisfy all requirements of VMs as their service-level agreements (SLAs). Due to a large number of VMs used by the enterprise and the demands for resources are dynamically changing, not all VMs can fit in the private cloud, and some VMs must be allocated to the public cloud to ensure the requirements of SLAs. Therefore, to maximize the enterprise's profit, we want to minimize the total cost of using the hybrid cloud while simultaneously satisfying the SLAs of all VMs.

As discussed in Section 3.2, the resources as well as the cost of private cloud including personnel, maintenance and electricity are fixed. So, to minimize the total cost of the hybrid cloud is to minimize the cost of the public cloud. The cost of the public cloud comes from the following three aspects. One is the hourly rate of resources used by VMs in the public cloud. The second is the fee of transferring data from the public cloud out. The last one is the penalty cost ($cost_{penalty}$) if any violations of SLAs happen. Thus, in

one period T such as in one month, the total cost can be the summation of these three terms as shown in Eq. (3.1).

$$cost_{tot} = \sum_{t}^{T} cost_{t,M_{pub}} + p_{tr} \sum_{t} tr\_out_t + cost_{penalty} \tag{3.1}$$

where $cost_{t,M_{pub}}$ indicates the total resource hourly cost of the public cloud at a period t based on the resource utilization of a set of VMs in the public cloud (M_{pub}). P_{tr} is the price rate of transferring data out of the public cloud in a one-month period. tr_out_t refers to the transfer-out amount of data at a period t of this month. Here we assume the cost of the public cloud is measured and charged hourly as we present in Section 3.2.

Since the utilizations of VMs are dynamically changing and the resources are fixed in the private cloud, to satisfy the SLAs of all VMs and to minimize the cost of using the public cloud, the resource utilizations of VMs in private and public clouds should be periodically monitored and some VMs should be migrated during a small period (e.g., 1 hour) to avoid any possible violations of SLAs. In this chapter, we plan to answer the following questions such that the total cost of this hybrid cloud can be minimized without causing any violations of SLAs of VMs: 1) How to do the migration, from private to public or from public to private? 2) When to trigger the migration? 3) Which VMs should be migrated?

To solve these problems, those most related studies can be categorized into two types of approaches. One type [43] (**Baseline#1** or **BL#1** for short) is to trigger VM migration from the private cloud to the public cloud when any resource utilization reaches to a predetermined threshold (named forward threshold or f-TH for short) (e.g., 90%). Based on this overflowed resource, the VMs are sorted based on their utilization of this particular resource. After that, the VMs will be put into a migration queue one by one starting from the VM, which uses the least amount of resource to the VM using the largest amount of resource until the total utilization of the overflowed resource of the remaining VMs is smaller than f-TH. Finally, the VMs in the migration queue will be migrated from the private cloud to the public cloud. Moreover, if overflow happens on multiple resources, following similar steps as mentioned above, the VMs are sorted based on each overflowed resource independently. Then, the VMs are put into different migration candidate queues (one per overflowed resource) until their required

corresponding resources of the remaining VMs are below the threshold in the private cloud. The VMs that appeared multiple times in these queues have a higher priority to be selected for migration. The rest of the VMs will be selected following a round-robin manner from different queues until the required resources of the remaining VMs in the private cloud are all below the threshold. For the migration direction from the public cloud to the private cloud, it also has a threshold (named backward threshold or b-TH for short) (e.g., 70%), which is smaller than f-TH. The migration will be triggered from the public cloud to the private cloud only when all resource utilization in the private cloud are lower than b-TH. Selecting VM migration candidates follows a similar aforementioned process, which first ranks VMs based on the utilization of different resources independently and selects VMs with a round-robin manner until all required resource utilization of the remaining VMs are below b-TH.

The other type of approaches [44, 45, 46, 47] (**Baseline#2** or **BL#2** for short) focuses on optimizing VM allocation. The approach follows the same procedure as in **BL#1** to trigger the migration when any resource utilization reaches a predetermined threshold. As for how to migrate/allocate VMs and minimize TCO, the problem can be formulated to fully utilize the resources in the private cloud. Thus, the issue becomes to maximally fit VMs into the private cloud such that the unused resources in the private cloud are minimal. To solve the issue, we use the scheme in [47], which summaries all resources by a combined factor (called fitness). We can give all three resources the same weights and then sort the fitness of all VMs. Then, the VMs will be put into the private cloud queue one by one from the VM with the largest fitness factor to the VM with the lowest fitness factor. If any resource is overflowed by allocating the current VM, we just skip the VM until going through all VMs. In the end, we have heuristically maximized the fitness factor in the private cloud and the remaining VMs will be put into the public cloud.

3.4 Algorithm Design

We assume that the cost of running VMs in the private cloud is lower and pretty much fixed. Thus, it is important to maximize the usage of the private cloud and reduce the cost of running VMs in the public cloud. The pricing model in the public cloud follows

Table 3.1 and Table 3.2 in Section 3.2.2. We consider three types of resources (i.e., CPU, memory, and storage) needed by each VM in this chapter. That is, if a VM is allocated with the required resources of CPU, memory and storage, it will satisfy its SLA. We further assume that there is enough network bandwidth to migrate VMs.

In this section, we start to introduce the proposed E-VM to migrate VMs between private and public clouds. The E-VM consists of two major types of components. One is VM-predictors to determine if any VMs need to be migrated to the other type of cloud. The VM-predictor's purpose is to prevent any resource overflow in the private cloud and reduce costs in the public cloud. The other is VM-selectors that use heuristic algorithms to select a set of VMs to be migrated to ensure that each VM has enough resources to satisfy its SLA.

3.4.1 Overall Structure

The major issues of a hybrid cloud are to determine when to trigger VM migrations and which VMs need to be migrated. There are two migration directions: from private to public cloud, and from public to private cloud. The reason to migrate VMs from the private to public clouds is that the total demand of resources by the current VMs in the private cloud is or will be overflowed, thus violations of the SLAs of VMs may occur. Migrating a set of selected VMs to the public cloud will release some demand of resources such that the resources of the private cloud are adequate for the remaining VMs, thus avoid any SLA violations. Meanwhile, the public cloud can provide an elastic amount of resources for VMs to satisfy their requirements with some extra cost based on the amount of resources is used. Therefore, the migration from private to the public cloud is necessary when the demand for resources by VMs is increased due to bursty type of workloads.

Moreover, the purpose of the migrating VMs from the public cloud to the private cloud is to minimize the TCO. The cost of running VMs in the public cloud is much higher than that in the private cloud. Thus, it is always necessary to save costs by migrating VMs back to the private cloud if there are enough available resources and the resources are underutilized in the private cloud. However, a difficult part is that if the private cloud stays close to the full utilization of its resources, it has a high probability of violating the SLAs of some VMs even with a small burst of demand on resources. The

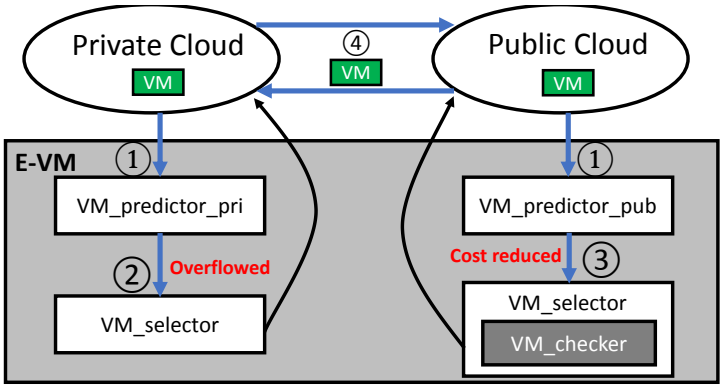

Figure 3.1: Overall structure of E-VM.

SLA violations will induce some penalty for the enterprise, which increases the TCO as well. On the other hand, if we keep a large margin between available resources and expected used resources in the private cloud, it means that more VMs have to run in the public cloud. Although this is safer to avoid any SLA violations, it also increases the cost when running too many VMs in the public cloud.

As shown in Figure 3.1, the E-VM follows four major steps for the VM migrations between private and public clouds. 1 uses VM-predictors to predict the future utilization of different resources of VMs in either private or public clouds according to the historically collected information. If the overflow of demanded resources is detected, in 2 VM-selector is used to select VMs to migrate from private cloud to public cloud to ensure the SLAs of all VMs are satisfied. If a certain cost-reduction condition is satisfied, it will advance to 3. That is, VM-selector will identify a set of VMs to be migrated from the public cloud to the private cloud. The private and public clouds have different criteria to select migration candidates, detailed in the following subsections. The last step, 4, is to do a live VM migration of the selected migration candidates between private and public clouds. In this chapter, we assume one of the existing VM live migration techniques can be used [48, 49, 50, 51, 52, 53, 54, 55] and there is no SLA violation happened according to the live migration. The following subsections provide the details of the E-VM scheme.

Algorithm 6 VM-selector: migration from private to public

```
1: procedure VM selection - single factor overflow
2: // **FA1 is the overflowed factor/resource **//
3:     OF = max(LP_mean, SP_max) − private_FA1 //overflow value on FA1
4:     Reset cur_FA1, cur_FAs, mig_VM and i_prev
5:     for i in VMs do
6:         Comb[i] = coef1*i.FA1 + coef2*i.FA2 + coef3*i.FA3
7:     end for
8:     vm_rank ← ranking Comb with descending order
9:     for  i in vm_rank  do
10:        if OF > cur_FA1 then
11:            mig_VM.append[i]
12:            cur_FAs = cur_FAs + α*i.FA2 + β*i.FA3
13:            cur_FA1 = cur_FA1 + i.FA1
14:            i_prev = i
15:        else
16:            if OF ¡ exchange i_prev and i plus mig_cand) and
17: i_prev.FAs ¿ FAs in i plus mig_cand then
18:                mig_VM.remove(i_prev)
19:                mig_VM.append(m for m in mig_cand)
20:                mig_VM.append(i)
21:                mig_cand.clear()
22:                i_prev = i
23:            else if OF > exchange i_prev and i plus mig_cand) then
24:                mig_cand.append(i)
25:            else
26:                mig_cand.clear()
27:            end if
28:        end if
29:    end for
30: end procedure
```

3.4.2 Migration from Private to Public Cloud

As indicated in Figure 3.1, there are two steps to decide when and which VMs to migrate. For the resource overflow prediction, the VM-predictor collects and records the utilization of three resources (i.e., CPU, memory, and storage) in the private cloud for each observation period (e.g., every 15 mins). To accurately predict the future resource utilization, a technique called Autoregressive Integrated Moving Average (ARIMA) [56] is used in this chapter. ARIMA is widely used for time-series forecasting problems based on non-stationary data. Some other time-series forecasting models such as Long short-term memory (LSTM) [57] can also be used to replace the ARIMA predictor. However, the LSTM model may need more datasets and take a longer time to predict. Thus, we decide to use the ARIMA model as the predictor.

The VM-predictor-pri predicts the overall utilizations of three resources for the near future (next observation period) in the private cloud. If the demand for any resources is likely to be overflowed, the VM migration must be triggered to avoid any SLA violations. Since the demand for resources may be bursty and unpredictable, the ARIMA model's prediction may not be accurate. We use two methods to mitigate the inaccurate prediction by the VM-predictor-pri. We add a Confidence Interval (CI) to the ARIMA model and use the upper bound as the predicted value. By doing so, the upper bound gives us a margin to tolerate some bursty workloads. Moreover, we use VM-predictor-pri to predict two different numbers of future points. One is a shorter length with few near future points (e.g., next four future points), and the other is a longer length (e.g., sixty future points). According to the ARIMA model, the shorter period can give more accurate results. The longer period is used to predict the trend of resource utilization. So, we use the maximum value of the following two predicted values to determine whether the migration should be triggered or not (i.e., whether the demand for any resource may be overflowed or not): one is the predicted maximum value by shorter period (SP_{max}) and the other predicted average value of the longer period (LP_{mean}). If any of these two values are higher than the resource capability in the private cloud, the migration will be triggered and the process advances to 2 in Figure 3.1.

At 2, the VM-selector selects an appropriate set of VMs for migrating to the public cloud. Since the three resources are considered, we have two VM selecting strategies. One is that only a single resource will be overflowed, called **single-factor overflow**. In this type, the VM-selector should find the VMs with high utilization of the overflowed resource but with low utilization of the other two resources. The other is that two or three resources will be overflowed in the private cloud, called **combined-factor overflow**. In this case, the VM-selector should pick up the VMs with high utilization on all these overflowed resources (factors) and make sure the utilization of non-overflowed resources is as low as possible.

We first describe how Algorithm 6 handles the single-factor overflow. The overflow value (OF) is computed by the predicted value ($max(SP_{max}, LP_{mean})$) minus the resource capacity (private_FA1) of the overflowed resource/factor FA1 in the private cloud. Then, the Comb value is computed for each VM based on its weighted resource utilizations. If one resource (factor) is overflowed (FA), its coef is assigned with

$10 * FA/Phy\_FA$ (The coefficient 10 distinguishes the overflowed and non-overflowed factors and also makes the non-overflowed factors considered in the scheme). Otherwise, it is assigned with FA/Phy_FA (Phy_FA is the resource capacity of private cloud for resource FA). We then ranked the VMs in the private cloud based on their Comb values with a descending order (Line 7). After that, the VMs are put into a migration pool (mig_VM) until the accumulated utilization of the overflowed resource by VMs in the pool (cur_FA1) is larger than OF.

Moreover, to minimize the accumulated utilization of other resources (non-overflowed) for the VMs in the pool as low as possible. We use a heuristic algorithm to accomplish this. As indicated in Lines 9-20, a parameter (cur_FAs) indicates how much utilization of the other resources will be migrated. If OF becomes smaller than FA1 in mig_VM, we start to check if there are new VMs that have smaller demands for FA2 and FA3 than the last inserted VM in the mig_VM. The weights α and β reflect the cost overhead of the other two resources in the public cloud. For example, in the public cloud, if only 5% margin for FA2 advances to the next price level with a cost increase of \$0.2/h and 10% margin for FA3 with an extra \$0.3/h cost, then $\alpha = 0.1$ for FA2 and $\beta = 0.3$ for FA3. The reason is that we do not want to migrate VMs with the high utilization of other resources to the public cloud resulting in a higher unnecessary charge of other resources. The coefficients can help distinguish the cost influence of different resources in the public cloud.

Similarly, for combined-factor overflow, the VM-selector should pick up the VMs with high utilization on all these overflowed resources and ensure the utilization of non-overflowed resources is as low as possible. First, we need to make sure that the OFs of all overflowed resources are smaller than that in the mig_VM. In other words, after migration, there is no overflow for any factor/resource in the private cloud. The selection scheme prefers to pick up the VMs having high utilization on those overflowed resources (use Comb value in Line 6 of Algorithm 6). This is because some VMs have strong correlations between different resources. For example, one VM with high CPU utilization may also have high memory utilization. So, choosing such a VM to migrate can reduce the utilization of multiple overflowed resources. We use a similar heuristic algorithm to replace the VMs with high utilization of non-overflowed resources in the mig_VM with VMs having low utilization of non-overflowed resources.

3.4.3 Migration from Public to Private Cloud

The purpose of VM migration from the public to the private cloud is to reduce the public cloud's cost. If too many available resources are not used in the private cloud, we should consider migrating some VMs from the public cloud to reduce the cost of the public cloud. However, this needs to be done carefully since there is a higher possibility of violating the SLAs of some VMs if the resources in the private cloud are almost fully utilized. Moreover, we need to avoid ping-pong migrations between public and private clouds. That is, we do not want to migrate some VMs between public and private clouds back and forth in a short duration. A ping-pong migration may cause a high charge since transferring data out from the public cloud is not free as seen in Table 3.2. Similar to the migration from private cloud to the public cloud, there are two steps to follow, but with different conditions and a different selection strategy.

We use the same predictor (named VM-predictor-pub in the public cloud) and prediction model to estimate the future resource utilization of the public cloud. Unlike the migration from the private cloud to the public cloud, the following two conditions are used to trigger a migration. One (**Cond#1**) is that the migration can make the charge of public cloud usage reduced by at least one pricing level according to the public cloud pricing model. Since the pricing model, as shown in Table 3.6, is discrete, migrating some VMs back to the private cloud may not reduce the overall cost. The other condition (**Cond#2**) is that the private cloud has enough unused resources to accept the migrated VMs. To avoid the ping-pong migration, the upper bound of the confidence interval (CI) is used for predicting the utilization of resources. We take CI = 90% as a default value in all evaluation experiments.

The VM-selection process is shown in Algorithm 7. First, the E-VM predicts how much resource utilization should be reduced in the public cloud such that the pricing rate can be degraded to a lower level (Lines 22-25). Then, based on the predicated resource utilization of public and private clouds, we check if the current situation satisfies the **Cond#2** (Line 8). After that, we use the proposed heuristic algorithm to put the VMs into vm_rank based on their Comb values. Similar to Algorithm 6, $coef$ values are computed from pub_FA/Phy_FA (Phy_FA is the resource capacity of private cloud for the factor FA). Starting from the highest Comb value in vm_rank, we put VMs in the mig_VM pool just before violating the condition (**Cond#2**). Moreover, the **Cond#1**

Algorithm 7 VM-selector: migration from public to private

```
1: procedure VM SELECTION
2:     Reset cur_cpu, cur_mem, cur_sto, mig_VM and i_prev
3:     Obtain available private cloud resource margins: mgn_cpu, mgn_mem, mgn_sto
4:     pub_cpu, pub_mem, pub_sto = pub_level()
5:     for i in VMs do
6:         Comb[i] = coef1*i.CPU + coef2*i.Mem + coef3*i.Storage
7:     end for
8:     vm_rank ← ranking Comb with descending order
9:     if mgn_cpu ¿ pub_cpu and mgn_mem ¿ pub_mem then
10:        for i in vm_rank do
11:            if i not in mig_VM and mgn_cpu > cur_cpu + i.cpu and mgn_mem > cur_mem + i.mem
   and mgn_sto > cur_sto + i.sto then
12:                mig_VM.append[i]
13:                cur_cpu = cur_cpu + i.cpu
14:                cur_mem = cur_mem + i.mem
15:                cur_sto = cur_sto + i.sto
16:            end if
17:        end for
18:    end if
19:    if pub_cpu ¿ sum(i.cpu in mig_VM) and pub_mem ¿ sum(i.mem in mig_VM) then
20:        mig_VM.clear()
21:    else
22:        for i in mig_VM do
23:            if cost_mig(i) > cost_normal(i) then
24:                mig_VM.remove(i)
25:            end if
26:        end for
27:    end if
28: end procedure
29: end
30: procedure PUB_LEVEL()
31:    cpu_util = max(LP_mean.cpu, SP_max.cpu) - level_cpu
32:    mem_util = max(LP_mean.mem, SP_max.mem) - level_mem
33:    return pub_cpu, pub_mem
34: end procedure
```

(Lines 15-16) checks if the cost is reduced based on the VMs in the migration pool. If not, the migration pool will be cleared and no VMs will be migrated to the private cloud. Finally, mig_checker (Lines 18-22) checks the cost (called transferring out fee) of migrating these VMs out the public cloud. If the cost is higher than the one-month normal usage fee, these VMs will be kept in the public cloud. The reason for using a one-month charge is that the transferring out fee is based on a monthly rate.

3.4.4 Overhead Discussion

In this subsection, we discuss the overhead of applying the E-VM scheme in a hybrid cloud environment. The overheads of the E-VM scheme are mainly extra space and computing time. One is the space overhead from recording VM resource utilization information for both private and public clouds. The information includes the history of the overall utilization of CPU, memory, and storage in the private and public clouds. Assume we use the most recent 25 hours (100 historical monitored points) to predict future one hour and four hours (future four and sixty points) and data are stored with float format (4 bytes). Therefore, total amount of history information is 100 * 3 (resources) * 2 (clouds) bytes = 600 bytes. For each VM, we record their most recent 4 points. Then, we have 12*N bytes (N is the total number of VMs). So, the overhead of recorded information is really small (400 VMs only need the space overhead of about 5.5KB).

The other overhead is the time required from building and executing a time-series prediction model. We investigated the overhead in a system with Intel(R) Xeon(R) CPU E5-2620 v3 2.4GHz processors. We use the Python3 with the statsmodels library. The execution time of building and executing an ARIMA prediction model is about 0.107s. Compared to the monitor time interval (15 minutes in this work), the execution time is tolerable.

Table 3.4: Private cloud resource configurations for the scenarios with different numbers of VMs

# of VMs	vCPU	Mem(GB)	Storage (TB)
40	16	32	1
100	32	96	16
400	128	512	64

3.5 Experimental Results

3.5.1 Environment Description

We captured a mixture workload with 16 applications running on 400 VMs (i.e., 400 individual services) in a hybrid system of the HPE Lab. The applications are described in Table 3.5. The peak utilizations of CPU, memory, and storage of each VM is captured for every 15 minutes. The lengths of those traces are 1250 hours. We then built an in-house simulator based on different migration schemes to calculate their overall cost ($). To investigate the scalability of this work, three scenarios with different numbers of VMs (40 VMs, 100 VMs and 400 VMs) are considered in these experiments. The scenarios of 40 VMs and 100 VMs select VMs following a round-robin manner from 16 applications. Correspondingly, we set up several private cloud configurations for these three scenarios, as shown in Table 3.4. The pricing model follows Table 3.2, Table 3.3 and Table 3.6 for migration cost, penalty cost and normal cost, respectively.

3.5.2 Overall Comparison

In this subsection, we make comparisons between the proposed scheme and the two baselines described in Section 3.3. Baseline1 (BL#1) is based on independent resources to select VMs as migration candidates. Baseline2 (BL#2) uses a heuristic algorithm to optimize VM locations when the migration is triggered. By default, the trigger migration threshold is set to either 70% or 90% for these two baselines, denoted by BL#1-70, BL#2-70, BL#1-90, and BL#2-90, respectively. As shown in Figure 3.2, we can find that the proposed scheme can reduce the overall cost by 1.4x - 4.6x, 1.4x - 11.6x, and 6.1x - 14.6x for the workloads of 40 VMs, 100 VMs, and 400 VMs, respectively. To analyze the results, we breakdown the cost into three categories (*Usage*, *Migration* and *Penalty*). *Usage* refers to the cost of normal utilization of VMs running in the

Table 3.5: Configurations of virtual machine workloads with running different applications

Applications	VM Type	IO Pattern	# of VMs	Avg. vCPU	Avg. memory (GB)	Avg. storage (GB)
CephS3	Mission Critical	Sequential	30	0.08	0.12	62.85
MySQL_Prod	Mission Critical	Random	30	0.98	7.76	219.06
MySQL_Test	Test and Dev	Random	50	1.76	4.53	36.28
Exchange_Tx	Mission Critical	Mixed	30	0.68	0.78	495.89
Intranet_ProjTx	Business Critical	Sequential	40	0.73	0.83	171.38
Internet_ProjTx	Business Critical	Random	30	0.50	1.27	167.93
CouchDB	Staging	Random	20	0.55	1.61	47.30
VDI	Business Critical	Mixed	30	0.34	0.44	36.90
VDI_Europe	Business Critical	Sequential	30	0.48	0.07	202.96
Bak_NonTradApp	Backup	Streaming	20	0.52	1.69	60.50
VDI_Asia	Business Critical	Random	20	0.32	0.05	243.53
Bak_TradApp	Backup	Sequential	30	0.28	0.10	204.42
SAP_Ariba_Prod	Mission Critical	Random	10	0.90	1.52	2157.16
SAP_Ariba_Stage	Staging	Sequential	10	0.08	0.12	25.35
SAP_Ariba_Test	Test and Dev	Random	10	0.64	2.00	16.02
HDFS_Output	Test and Dev	Sequential	10	0.24	0.02	47.24

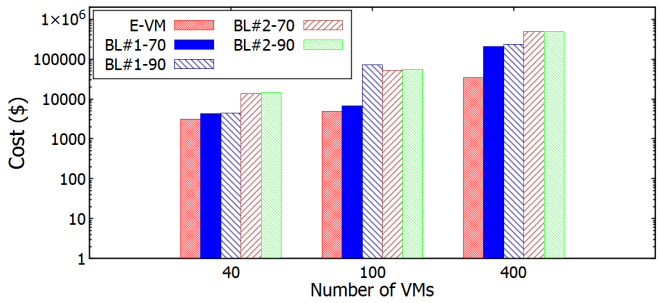

Figure 3.2: Overall cost comparison with different numbers of VMs.

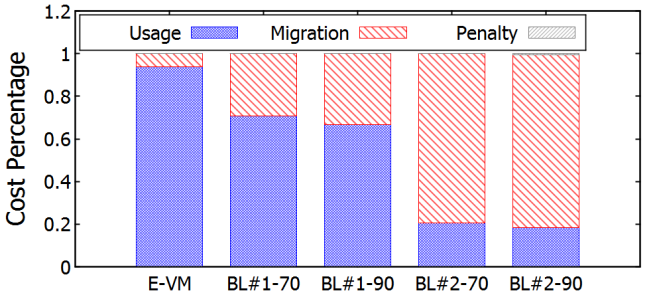

Figure 3.3: Percentage of different costs with 40 VMs workload.

public cloud. *Migration* refers to the migration cost when egressing out from the public cloud to the private cloud. *Penalty* demonstrates the penalty cost when SLA violations happen in the private cloud. The percentage costs of these three categories are indicated in Figure 3.3. We can find that the proposed E-VM achieves more than 90% normal usage cost, and the cost of combining migration and penalty is less than 10%. BL#1 and BL#2 get much larger migration cost (25%+ and 75%+, respectively) than E-VM. The reasons are threefold: 1) The static threshold cannot provide an accurate prediction of utilization trends and thus it potentially causes a higher migration overhead between private and public clouds. 2) The combined resource consideration in E-VM takes care of the potential correlations between different resources (e.g., CPU and memory) and correctly selects VMs to fit in either private or public clouds. It fully utilizes the private cloud resource and also potentially reduces the number of migrations triggered. 3) The migration checker filters out some VM migration candidates by comparing the cost between staying in public and egressing out from the public cloud to the private cloud.

Table 3.6: Pricing models of Amazon[1], Microsoft[3] and Google[4]

	Unit	vCPU	Memory (GiB)	Cost ($/h)	Storage ($/GB-month)
Amazon	m5.large	2	8	0.096	0.1
Microsoft	n1-std	1	2	0.036	0.075
Google	Av2	1	3.75	0.0475	0.17

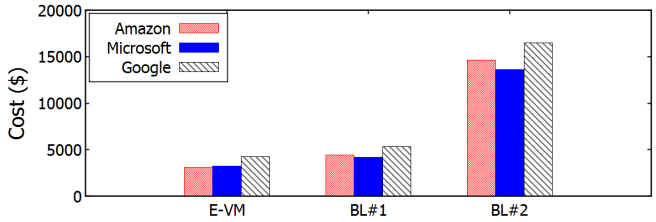

Figure 3.4: Overall costs by using different public clouds with 40 VMs workload.

As a result, it prevents ping-pong migrations between private and public clouds, thus reducing the overall cost.

3.5.3 Different Cloud Pricing Models

In this subsection, we investigate the overall cost of these schemes when using different cloud platforms. As shown in Table 3.6, we pick three similar general-purpose instances from Amazon Web Service, Microsoft Azure, and Google Cloud as our resource units. Following the discussion in Section 3.2.2, the resources in Table 3.6 are basic units, and the total amount of resources in the public cloud can be scaled based on how many resources VMs will use. The pricing models of transferring VMs out of clouds are following Table 3.2.

The total cost of different platforms with running three different workloads (40 VMs, 100 VMs and 400 VMs) are shown in Figure 3.4, Figure 3.5 and Figure 3.6 respectively. First, the proposed E-VM scheme always gets the least cost compared to the two baselines no matter what platforms we are using. The reasons are described in the previous subsection. To horizontally compare different platforms based on the pricing model in Table 3.6, Google obviously obtains the highest cost among three cloud platforms with running 40, 100, and 400 VMs for all three schemes. The reason is that Google has a much higher storage price rate (1.7x - 2.3x higher) and a higher

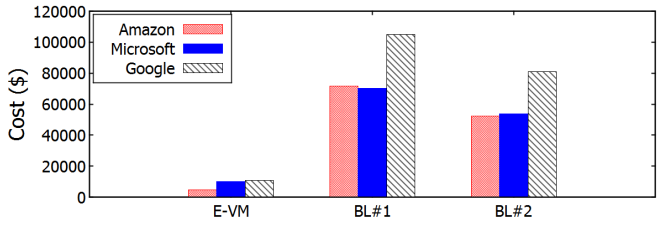

Figure 3.5: Overall costs by using different public clouds with 100 VMs workload.

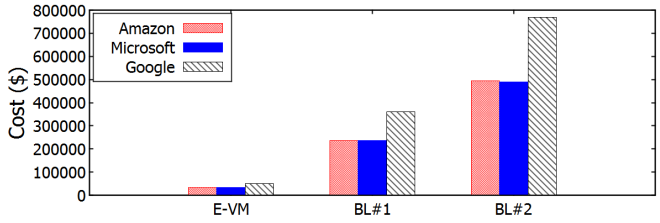

Figure 3.6: Overall costs by using different public clouds with 400 VMs workload.

cost rate of transferring data out (12% higher under 10TB) than those from Amazon and Microsoft. To compare between Amazon and Microsoft, they achieve similar cost. For some cases like the workload of 40 VMs, Amazon is higher than Microsoft, while Amazon obtains a lower cost in 100 VMs for E-VM and BL#2. To figure out a deeper understanding, we can normalize the cost to one vCPU and one GiB memory cost. Therefore, Amazon has \$0.048/vCPU and \$0.012/GiB memory, while Microsoft has \$0.036/vCPU and \$0.018/GiB memory. Obviously, memory-intensive workloads are better to use Microsoft or Amazon platforms, while CPU-intensive workloads prefer to use the Microsoft cloud platform.

Lesson#1*, people can always find a similar pricing model in different cloud providers since the cloud providers try to use similar rates with their competitors. However, the little difference may give users a significant benefit based on the requirements of workloads.*

3.5.4 Individual Types of Applications

In this subsection, we run each type of applications individually in the hybrid cloud. As mentioned in Section 3.5.1, 16 types of workloads have different properties. We run them separately to see their total cost effect. The private cloud configuration follows the

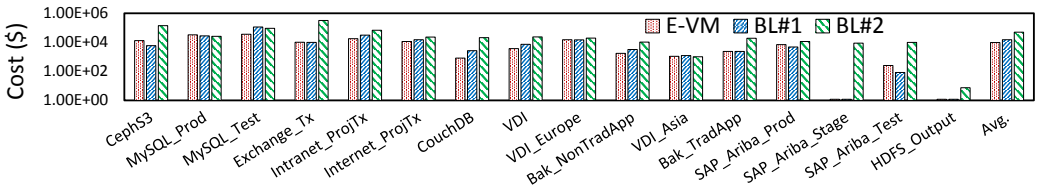

Figure 3.7: Total migration size and number of migrated VMs for two schemes.

40-VM configuration shown in Table 3.4. The price model follows the Amazon pricing model in Table 3.6.

As we see in Figure 3.7, in total 16 workloads, on average, the proposed E-VM achieves 36.3% and 80.7% lower cost than BL#1 and BL#2, respectively. Moreover, it outperforms the two baselines with nine workloads. E-VM and BL#1 achieve similar in four workloads (e.g., Exchange_Tx and SAP_Ariba_stage). BL#2 always obtains the highest cost due to a high migration overhead. Meanwhile, BL#1 has a lower cost than E-VM in three workloads (i.e., CephS3, MySQL_Prod, and SAP_Ariba_Test). The reason is that these three workloads have more frequent dramatic resource utilization changes and thus the predictor in E-VM cannot accurately predict the trend of utilization changes. As a result, a larger penalty is induced due to more resource overflows in E-VM and thus, incurs a higher cost than BL#1. For the mixed workloads discussed in Section 3.5.2, dramatic resource utilization changes of few workloads might be canceled or mitigated by mixing different workloads. Therefore, for those workloads of 40-, 100-, and 400 VMs, E-VM always outperforms the two baselines.

3.5.5 Effect of Separate Features

In this subsection, we discuss the effect of different individual features. Starting from BL#1, we add one feature each time until it becomes E-VM as described below:

[leftmargin=*]**BL#1:** is the baseline scheme which uses static threshold to trigger migration and consider each utilization separately. **+Dynamic:** adds dynamic utilization prediction based on BL#1. **++Combined:** considers correlation between factors/resources based on +Dynamic. **+++Mig:** uses a migration checker in the public cloud based on ++Combined, which is **E-VM**.

We compare the above four schemes with three workloads. The pricing model is the

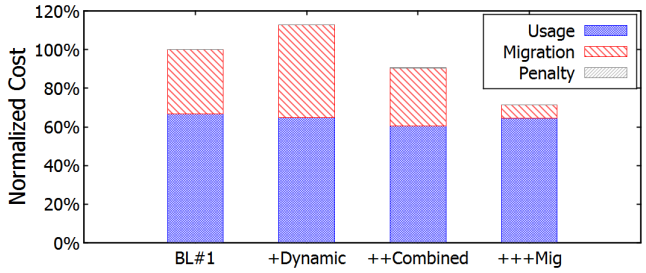

Figure 3.8: Total cost of individual schemes with the 40 VM workload.

same as that in Section 3.5.2. To conveniently see the cost reduction, the total cost of BL#1 is normalized to 1. From Figure 3.8, Figure 3.9, and Figure 3.10, we can find that in general the total cost is gradually decreased from BL#1 to +++Mig (E-VM). On average, +Dynamic obtains 19% lower cost than BL#1, ++Combined gets 16% lower cost than +Dynamic and +++Mig achieves 59% lower cost than ++Combined. There are some exception in Figure 3.8 and Figure 3.9. For example, +Dynamic gets a higher cost than BL#1. The reason is similar to the scenario of CephS3 in Section 3.5.4. Due to irregular changes of resource utilization, the utilization predictor inaccurately predicts the future utilization and triggers some unnecessary migrations between public cloud and private cloud. For another exception in Figure 3.9, when considering correlations between resources, we can migrate VM more efficiently and fully utilize private cloud resources, resulting in less migration overhead and less normal usage in the public cloud. However, when any resource utilization faces a burst, it induces a higher risk of violating SLAs in the private cloud. Therefore, the SLA violation penalty in this case is higher than the others.

Lesson#2*, based on the modern cloud pricings, the price strategy of cloud providers is to discourage users from migrating VMs out from public clouds.*

3.6 Real System Evaluation

The proposed scheme E-VM and Baseline1 are applied in a real system, and we made a comparison for these two schemes in terms of migration overhead using 1000 Virtual Machines with an average of 8 virtual CPUs per VM. At the beginning of the test

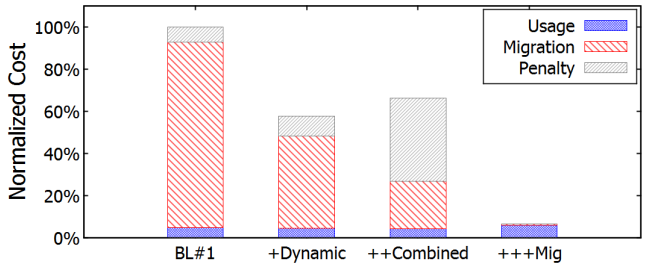

Figure 3.9: Total cost of individual schemes with the 100 VM workload.

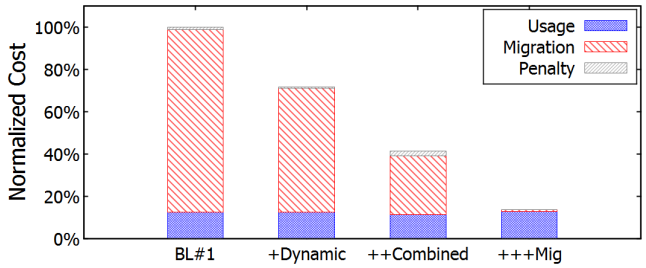

Figure 3.10: Total cost of individual schemes with the 400 VM workload.

cycles, 500 Virtual Machines were in the private cloud and 500 Virtual Machines were hosted in Amazon elastic services. The workload used was the clone copies of the Hewlett Packard Enterprise data center workload, which had replication set up with the production workload. This workload allows us to get the performance of a realistic workload as close as possible. The experimental environment had 100 HPE DL380 Gen 10 Servers. Each server configured with Intel Xeon Model, Gold 6248R Processor with 24 core 3.0GHz processors, 35.75 MB L3 Cache, 2 @10.4 GT/s UPI and 2933 MT/s DDR4 memory 1 TB per socket. Each Server is configured to host 4 Virtual Machines using VMware vSphere 6.7 U3. The Network Controller on each server, 10/25Gb 2-port LFR-SFP28 MCX4121A-ACFT Adapter. The Storage Controller on each server, P408i-a w/2GB cache 8 port modular Smart Array with 24 SFF SSDs in Front and 6 SFF SSDs in the rear.

First, we investigate the data movement in hybrid clouds. The less data is moved, the better it is from reducing the cost point of view because the data in and out is one of the important cost parameters in the public cloud. As shown in Figure 3.11, it is clear to see that the BL#1 model performs worse than the proposed scheme. Specifically, BL#1 migrates 4.9x more data (2.6x more VMs) than the proposed E-VM scheme. This

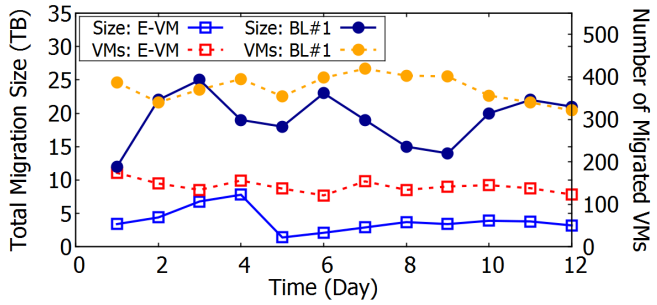

Figure 3.11: Total migration size and number of migrated VMs for two schemes.

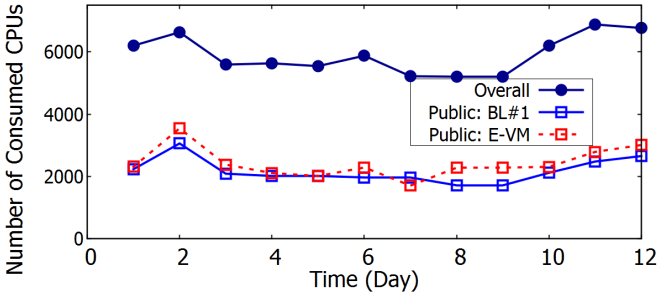

Figure 3.12: Overall CPU consumption (public+private) and the private CPU consumption of E-VM and BL#1.

is primarily because the simple decision model trying to keep a large number of VMs in the private cloud makes the algorithm aggressively move VMs from the public cloud to the private cloud. Also, this BL#1 algorithm more aggressively picks up VMs of low CPU utilization from the public cloud to migrate.

Moreover, one key observation from Figure 3.12 is with the number of the total number of CPUs used in the public cloud. The E-VM approach has worked better on all but one iteration against BL#1. On average, the E-VM scheme saves about 12% CPU consumption compared to BL#1. Another interesting observation is that E-VM tends to stack up the public cloud with the dormant CPUs from the private cloud. This tends to keep the count of virtual CPU cores less on the public cloud. In a hybrid cloud, this reflects less cost due to the pricing model of public cloud using the number of CPU cores and better utilization of resources in the private cloud.

3.7 Related Work

For the hybrid cloud's research topics, people mainly focused on three aspects of the hybrid cloud.

In the first aspect, the previous studies [46, 58, 59, 45, 60, 61, 62, 63, 64, 65, 66, 67, 68] mainly focused on task/VM allocation. For example, Liang *et al.*[46] proposed a cost-driven model for scheduling tasks/VMs in hybrid clouds by using a firework algorithm based algorithm. Their work mostly focused on scheduling tasks on VMs. VM allocation is only considered by the availability of resources in an environment with small-scaled physical machines. Finally, they minimized the cost from a user's perspective. H2-D2 [59] is an approach focusing on the problem of multi-objective VM reassignment for large and hybrid data centers. Multiple dimensions of the infrastructure optimization including the overhead of running IT infrastructure, reliability and migration were independently considered. However, they statically reassigned VMs to different clouds, which are not sufficient for handling dynamical workloads. Also, they did not consider the pricing model for a hybrid cloud environment.

Another aspect is that different roles in hybrid cloud set different objectives. The cloud service providers try to maximize their profit including minimizing their IT infrastructure cost and delivering proper prices [59, 45]. For example, Liu *et al.* [45] proposed a scheme to decide the resources in a private cloud and schedule the requests from users to private or public clouds. Also, they tried to decide the optimal prices for the public cloud service providers by using a Stackelberg game model. Therefore, this work is trying to solve the issues for public cloud service providers. The difference between individual users and cloud service customer (CSC) is whether they have their own private cloud. Normally, the enterprises as cloud service customers have their own private cloud and individual users also purchase services from cloud server customers such as HPE and Dropbox. For the individual users and CSCs, they tried to minimize their cost on the public cloud by fully utilizing the resources purchased from public clouds or in hybrid cloud [58, 46, 62].

The third aspect focuses on the pricing model of cloud environment. From the perspective of cloud providers, they try to set up proper pricing strategies for their products [1, 3, 45]. From a user perspective, they allocate resources based on different

pricing models to minimize their overall cost [59, 60].

In summary, most of the above work did not manage VM allocation and migration in a hybrid cloud from an enterprise point of view. They did not fully consider the dynamically changing workloads of VMs in hybrid clouds. They in general ignore the utilization correlations between different resources and lack a comprehensive consideration of pricing models.

3.8 Conclusion

In this chapter, we propose a smart migration scheme for hybrid clouds, called E-VM, to target on fully utilizing the resources in the private cloud and minimizing the cost of using the public cloud. The E-VM considers bi-directional migrations between private and public clouds. Two components (VM-predictor and VM-selector) are built in the E-VM to determine if there is any need for migration between private and public clouds and which VMs will be migrated to the other cloud. Moreover, the E-VM is designed based on one pricing model in Amazon Web Services and can be extended to different pricing models from other cloud providers. According to the experimental results, the proposed E-VM can reduce the total cost of using the public cloud by 1.4x - 14.6x compared to the existing VM migration schemes.

Chapter 4

Regulatory Compliance Considerations for Hybrid Cloud

4.1 Introduction

To meet the performance and the cost goals, workload and data mobility are the core underpinnings of Hybrid Cloud Infrastructure. But these goals must be accomplished while achieving the regulatory guidelines such as GDPR (General Data Protection Regulation) in EU and CCPA (California Consumer Privacy Act) in State of California. These regulations are in place to protect the privacy of consumers and protection of data sovereignty. Technologies such as Server Virtualization and Containers make workload mobility relatively easy to execute but they are built on Storage protocols in the pre-hybrid cloud era not designed to retain or enforce data locality, privacy.

Any Hybrid Cloud environment must meet the performance and cost goals for its services while responsibly protecting the privacy of consumers and sovereignty of the data. In this chapter we will, a) explore the practical implications of the regulations to the Hybrid Cloud environment, b) explore state of art technological steps taken by the top public cloud providers to meet these regulations and c) propose practical changes in the core Storage infrastructure layers to better enforce data locality and protect consumer privacy.

Workloads which operate on private datacenters are Private Cloud workloads whereas the workloads which operate on public clouds such as Microsoft Azure, Google Cloud

Compute, AWS and Alibaba are Public Cloud workloads.

Private cloud offers security, reliability and performance advantages. These private cloud advantages are attribute of the characteristics of the private data center architecture; it is harder to exploit man in middle attacks, infrastructure is custom built with reliability and performance as core design tenant. Public cloud offers ease of deployment, scale on demand and operating cost advantages.

The disadvantages of private cloud are they difficult to scale on demand, require high operating cost and have to manage legacy software stacks. The disadvantages of public cloud are security and privacy concerns, high cost to migrate data in and out of the public cloud.

With the advent of the server virtualization technology, we can build a workload which can seamlessly move workload from pubic to private and vice versa depending on the required workload characteristics. Such a hybrid cloud can retain and deliver the advantages of both public cloud and private cloud while minimizing their respective disadvantages.

There are significant data, privacy concerns which the service providers have to factor in while designing services for hybrid cloud. Government agencies have been increasingly concerned about the data privacy of the consumer in public cloud which is becoming a complicated issue with hybrid cloud, data mobility between public and private cloud. Public cloud providers own and control the hardware and the software stack on which the workloads are deployed. From such grounds up design applications it is easier to add metadata to build a framework to modernize security and privacy of the end user. But the private clouds have to factor in legacy application stack which are harder to adopt the regulations without re-writes.

European Union has taken the most comprehensive approach to make sure that the consumers data privacy is forefront in hybrid cloud. The two key attributes which are critical in GDPR are: a) user controls data and b) in hybrid cloud data can only travel to countries which has equal to or stricter data privacy laws than GDPR. The significance of these attributes is GDPR developed a framework which factors in need for workload/data mobility for hybrid cloud in a responsible way with consumer data privacy as the key core cornerstone to protect.

Our objective with this research is to propose a system architecture framework which

incorporates data privacy regulations in fluidity expected of hybrid cloud workloads from service providers point of view. The novelty of the proposal is that it doesn't require complete application re-write and thereby improving the changes of faster adoption for this framework.

In this chapter we first introduce the key foundational elements to GDPR and its implications to the system architecture, overview of the Storage Protocols and their theory of operations to highlight potential solutions to incorporate the needs for GDPR and location-based services with an illustration of how it could be used for the infrastructure management in the datacenter.

In the next section, we examine the state of art system architecture in industries which has to consider and deploy end user privacy. We considered a number of public cloud providers such as from Facebook, Google, Microsoft Azure and Amazon AWS. For this second section we decided to example Google's infrastructure among the other service providers since Google has published several academic papers regarding their infrastructure which help us examine and reference to advance the open system architecture.

In the fourth section, we propose a modernization framework for all Storage Protocols to factor in regulatory compliance and policy concerns. Application backward compatibility is key to adoption of the proposal in the open systems. There are two specific areas we investigated in this section. First, we propose embedding scalable access metadata in the data structures of the existing protocols. Our approach helps with the backward compatibility of the applications. Second, we noticed a major vulnerability in the Storage protocols with man in the middle attacks. This is understandable since most of the protocols were designed in the era prior to cloud. We propose an approach to address man in the middle attack based on the similar transformation and approach taken in the wireless authentication field.

Later in the experiment section we examine how both of our proposals described in earlier section perform. The research of expanding the Storage protocols to make them secure and incorporating the regulatory compliance to protect user privacy is a new field. We believe our proposals will also spin up many more research in making storage protocols more secure and regulatory compliant.

4.2 Background on GDPR, Storage Protocols and Location-based Services

GDPR [69] as a regulation was adopted on 14 April 2016 and became enforceable on 25 May 2018. The primary concerns of the regulations are two-fold: a) Data Protection and Privacy of the personal data by giving control to individuals over their personal data, and b) Addresses the transfer of personal data outside of EU (European Union) and EEA (European Economic Area).

The regulation also requires businesses, report data breaches to the national supervisory authorities within 72 hours if they have an adverse effect on user privacy. Financial penalty for violators could be as high as 4% of the previous year revenue for violating corporation.

GDPR has been the model for regulations across the world including CCPA [70] and regulations in countries such as Japan, South Korea, Brazil, Argentina and Kenya. GDPR is having a significant impact of changing the operating paradigm for application, data and storage architectures.

4.2.1 GDPR First Core Intent – User Controls the Data

One of the two core intents of the GDPR is to give the control of the data to the user. No personal data may be processed unless this processing is done under one of six lawful bases specified by the regulation; consent, contract, public task, vital interest, legitimate interest or legal requirement shown in Table 4.1.

The data management model however in data center is complex and it is incredibly hard to discern with certainty where all the copies of the data for a particular user reside to provide such granular control of the per object data management.

The application deployment model (and not the user) is the center of the data management in today's datacenter. The 4.1 below gives a typical scenario for how multiple copies of data can exist in the datacenter. These copies are encapsulated in the data format of application layers, operating environment format and various systems specific activities such as disaster recovery (DR systems) and Sequence of Full and Incremental Backups.

In such a model, there is no standard way to discern and confirm to the end user

Table 4.1: Typical 3-layer stack involved in the journey of data with multiple copies of itself

Deployment	Production/Test Deployment		Data Protection Deployment	
Application	ProdApp 1 to X	TestApp 1 to N	DR System	Backup
Data Management	VMDK 1 to X	Container 1 to N	Replicas 2 or 3	Full + Increments
Storage Layer	Block	File	Block & File	Object

by the system administration that user data processing was done under the six lawful bases specified by the regulation.

In this section, we will examine the 6 lawful bases for the data access and their implications on the system architecture [71, 72].

- Consent: Under the GDPR, one of the lawful ways to process the personal data is via an individual's unambiguous consent. The implication of the unambiguous consent is that there must be a clear affirmative action. This means that obtaining tacit consent for personal data is no longer valid and a one-time click consent also needs to be revalidated.

- Contract: This applies when the processing is related to the parties to a business, employment or administrative agreement and is required to maintain or fulfil the agreement. For example, this could be the processing of an employee's name, surname and photograph to produce a company ID card.

- Legitimate Interest: When the processing is necessary for compliance with a legal obligation of the data processor, as long as this is not overridden by the interests or rights and freedoms of the data subject, bearing in mind the reasonable expectations of the subject based on their relationship with the data controller. Therefore, although the controller may process the data without having obtained consent in virtue of their legitimate interest, the subject may also impose their rights and freedoms by exercising their right of opposition. We can see that among all these situations of lawful processing, the satisfaction of legitimate interests is the one that generates the most uncertainty as it can cover a wide range of concepts. Examples of cases in which the legitimate interests of the processor may apply are prevention of fraud, transmitting data within a group of companies, and transmitting data to ensure network security, use of what are known as "onboard

cameras" on vehicles, for the purpose of recording images as legal evidence in the event of an accident.

- Vital Interest: In this case, the processing is necessary to protect the vital interests of the data subject or another physical person. Examples of vital interests and public interest as those which require processing for humanitarian purposes (to control epidemics, for example) and situations of humanitarian emergencies, in particular in situations of natural and man-made disasters.

- Public task: The processing is required for the purpose of fulfilling a mission carried out in the public interest or in the exercise of public powers conferred on the processor. The GDPR stipulates that, both for the fulfilment of public interests and legal obligations, member states may maintain or introduce more specific provisions. An example of such interest would be, schools may obtain a central sex offenders' registry clearance certificate, which is required for everyone who works with minors.

- Compliance Requirement: One of the legal obligations that the data processor or the electronic trust service provider must preserve the data and documents for a period of 5 years. After which time the data can be safely destroyed or deleted with an explicit end user consent.

There are three types of access initiators [73, 74]: a) Personal user (PU), b) Data Processor (DP), c) Legal Processor (LP). The intent of the regulations is to protect the privacy of the user but there are pre-defined legal reasons for access to the data. In the Table 4.2 below we have articulated the key characteristics of all types of access.

Table 4.2: Implications of the various interests and key attributes affecting data management

Interest	Access Initiators	User consent	User notify	One time	Event based	Delete
Personal	PU	Y	Y	Y	N	Y
Contract	DP	Y	Y	Y	N	N
Legitimate	LP	N	Y	N	Y	N
Vital	LP	N	Y	N	Y	N
Public	LP	N	Y	N	Y	N
Compliance	DP	Y	Y	Y	N	Y

Data Processors are the applications providers managed or cloud hosted who manage the data for the personal users. Legal Processors are entities, which could be federal or state authorities. The user consent is not necessary for all the cases under which user data can be accessed however at some later point user can be notified about the reason for data access. Only the user and the Data Processor can delete the data assuming compliance after stipulated period of time. The Legal Processors can access the data on a specific event basis.

We can create a metadata structure for per access object with the following attributes mapped to the Table 4.2. The metadata structure could consist of the 8 key attributes which best captures the intent of the core tenant of the GDPR regulations. a) Interest ID, b) Action, c) Consent, d) Notified, e) Request Time, f) Duration, g) Event Bit and h) Delete Time.

4.2.2 GDPR Second Core Intent: Data travels only to countries with protections equal to GDPR

At the initial onset of the regulations, there were strict guidelines and regulations so that the data never leaves the country of where the data was originated. Example: Data generated in France could not leave or be accessed in Germany.

These restrictions became cumbersome and improbable to practically implement for following reasons: a) Data protection strategy for cloud providers may need the data to be present in more than one location and b) Legitimate business reasons for the data to be accessible from secure countries. In short, all provisions of data transfer shall be applied in order to ensure that the level of protection of natural persons guaranteed by GDPR is not undermined.

4.2.3 Storage Protocols and Theory of Operations

The storage protocols have not evolved with the advent of virtual machines and the containers. The existing block and file protocols which were designed with the client-server model for connectivity still continue to be the primary sources of data platform. So, all the security models which existed prior to the operating system virtualization and containerization continue to be challenge and new surface area of security concerns

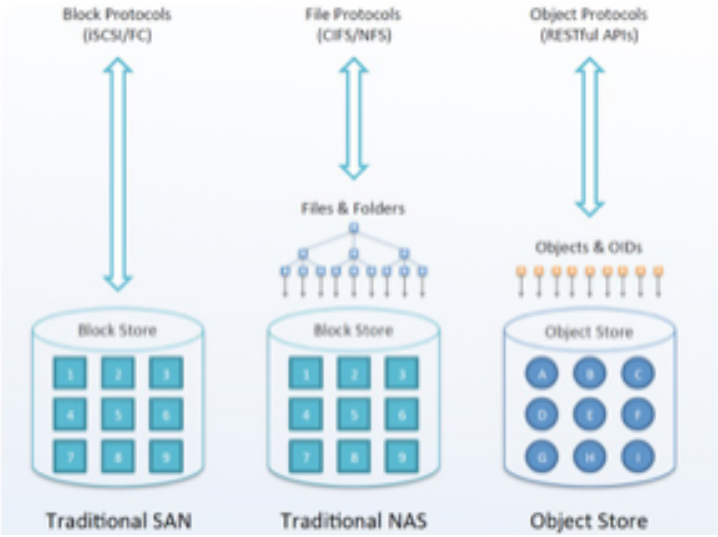

Figure 4.1: Illustration for the Block, File and object protocols

and policies have emerged which are not factored in these storage protocols, shown in Figure 4.1

Block Protocols provides the most granular form of data access is oldest and the most popular form for the storage protocols. SCSI (Small Computer System Interface) became a standard in 1986 long before internet era. Various transport layer such as Fibre Channel and iSCSI (internet-based SCSI), Ethernet based came much later but the core of this protocol to access, update and modify small blocks of storage randomly by the host operating principle has not changed.

File Protocols and ability to access files in a directory hierarchy emerged in the 90s and become popular due to the NFS (Network File Server), initially created by Sun Microsystems and CIFS (Common Internet File System), created by Microsoft. A remote host (client) can access the file from a central NAS (network attached storage server). The security models of NAS were much more robust than the block protocols and for the first time, the user level permission were first introduced for data management.

Object Protocols are contemporary protocols which are based on key value pair. The entire data per key is stored as single blob. This blob of data can be of arbitrary size and access to this data a query from the client side to access the blob of data associated with the key. Object Store are generally used in the large-scale repository and not

remotely accessed via protocols such as Amazon S3 and OpenStack Swift Object API. The comparisons are shown Table 4.3

Table 4.3:

	iSCSI	NFS/CIFS	S3
Device Authentication	iSNS (iSCSI Naming Server)	LDAP (Lightweight Directory Access Protocol) for NFS Active Directory for CIFS	Regions or Zones are designated areas where policy based multiple copies retain
End User Authentication	Device Level	Access Control List on per user or directory level	Access Control List for object or group of objects
Application Use Cases	Virtualization, Boot Devices and Databases	Virtualization, Containers and Unstructured data	Large data store mostly write once read many type
Location Awareness	No	No	Limited and not propagated to host or application levels

Theory of Operations and Compare for the three Storage protocols:

The iSCSI protocol is a point-to-point storage protocol connecting iSCSI initiator (an iSCSI client) to iSCSI target (an iSCSI storage device). The iSCSI client recognizes iSCSI target through initiator node name called iQN. In case of iSCSI, there is an optional server which keeps a list of all the available iSCSI targets, called iSNS. An iSNS server is responsible for informing iSCSI client which iSCSI targets are available on network, grouping iSCSI client to their correct domain sets and informing iSCSI clients which security aspects to use to associate the targets.

All the major operating systems have iSCSI initiator driver bundled as part of the core operating system. Because iSCSI data access is at a much granular layer at each block level of say 4K, 8K, 32K, etc. hosted on the LUN, there is not even context to assign a per application or a per user level authentication mechanism. This poses a significant security risk, if the iSCSI LUN is compromised then the entire data is visible to the attacker through any and all virtual machine and containers which is connected to the storage device.

The major iSCSI security issues are, a) iQN values are trusted, this is spoof-able,

vulnerable to man in the middle attack and values can be brute-forced. b) The iSCSI protocol in its core is clear text protocol, although recent version of iSCSI can make use of Kerberos, c) iSCSI use Challenge Handshake Authentication Protocol (CHAP) is a network login protocol that uses a challenge-response mechanism. The CHAP based authentication is used to restrict iSCSI access to volumes and snapshots to hosts that supply the correct account name and password (or "secret") combination.

The iSCSI protocol is not application location aware, there is no notion in the protocol about the origin for the iSCSI target.

NFS protocol is use today predominantly are either, NFSv3 or NFSv4. NFSv3 has limited security model. In this version there was no server authentication, and the client authentication was limited to defining a list of authorized clients on a NFS server. Systems are listed by hostname or IP address. NFS server are responsible for validating the source IP address, but there were no strict implementation verifying that the IP address claimed by the client were indeed the IP address. NFS v3 protocol made use of RPC (Remote Procedure Calls), the NFS server explicitly trusted the UIDs and GIDs provided by the underlying systems.

NFSv4 was a marked improvement over the NFSv3 protocol. However, NFSv4 is still considered very complex to deploy and NFSv3 still remains the most widely used NFS protocol. In NFSv4, the user authentication is done through Kerberos and the Server authentication is provided by LIPKEY (Low Infrastructure Public Key).

The NFS protocol like iSCSI protocol is not application location aware, there is no notion in the protocol for the NFS clients.

Object Protocol is contemporary storage protocols is simple in structure and designed for high latency internet connections. S3 is the de-facto standard in object protocol. The protocol re-uses the http authentication and authorization mechanism, and objects are created, stored and deleted using simply put and get commands served through the http protocol.

The objects stored from the object protocol are stored in buckets. Buckets are just a simple collection of objects of similar pre-defined domain/sets into a single logical collection. The access to the buckets is through access control list. For data availability, objects can be dispersed on multiple buckets in multi-datacentre. The metadata in the object protocol are easily extensible and because of its use of simple REST interface it

is popular to store a large repository of data.

Most common object protocols don't have the ability to update in place a particular file or an object. So, for all practical purposes, the objects in the object store are immutable. This is much different that the block or the file protocols where the objects can be updated or re-written in place.

The object protocol has some degree of application awareness, but it is used to make sure the data is highly available. Each object is replicated in to 2 or more than 2 copies for data availability reasons. The awareness in this case is for placement of the data in multiple data centre for high availability of access and not security of access.

The object protocols in that regard are more application aware than either the NFS protocol or iSCSI protocol however, it doesn't have notion of enforcing at the application level to make sure the objects are not accessed by applications outside of its granted privileges.

4.2.4 Introduction to Location Based Services

Location Based Services can be looked at two levels: a) Within the datacentre and b) Across the datacentre. Within the datacentre, wireless access point can be used and across the datacentre, we can use the location-based services available through cloud providers. With the advances in wireless access point [75] [76], it is possible to pinpoint the location of the rack and the servers in the rack to assign location.

Figure 4.2 illustrates a real live example of the asset tag which is located with the datacentre and Figure 4.3 below illustrates how we can get the entire datacentre wired with the location for creating the end-to-end location awareness from app and host operating system to the storage protocols. All the information can be programmatically access via well-defined API from the access points. The triangulation services are also available through the access points for pinpoint accuracy.

Across the datacentre, pioneered by the consumer business GPS and location-based services are readily available. Below is a quick snippet of the code on getting the latitude and longitude data in this case from Android API [77] [78] as in the following example.

```
mGeofenceList.add (new Geofence.Builder ()
    // Set the request ID of the geofence. This is
```

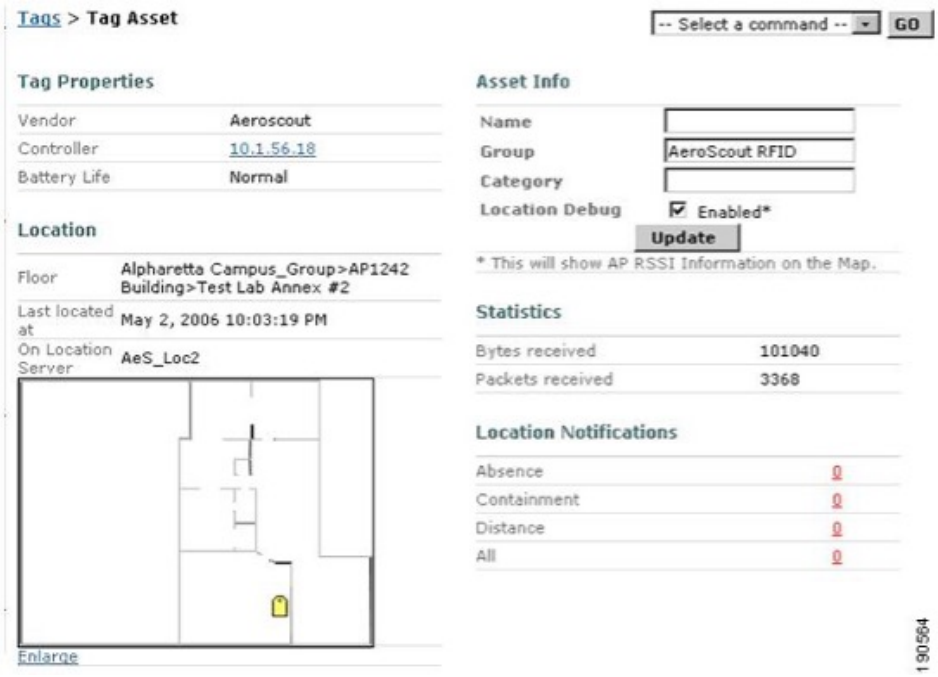

Figure 4.2: Asset Tag in the datacentre with floor plan using wireless access points

```
    // a string to identify this geofence.
  .setRequestId (entry.getKey ())
  .setCircularRegion (entry.getValue ().latitude,
      entry.getValue ().longitude,
        Constants.GEOFENCE_RADIUS_IN_METERS)
    .setExpirationDuration (
            Constants.GEOFENCE_EXPIRATION_IN_MILLISECONDS)
    .setTransitionTypes (Geofence.GEOFENCE_TRANSITION_ENTER |
        Geofence.GEOFENCE_TRANSITION_EXIT)
    .build ());
```

The location information both within and outside the datacenter can be easily obtained programmatically through the well-defined API sets.

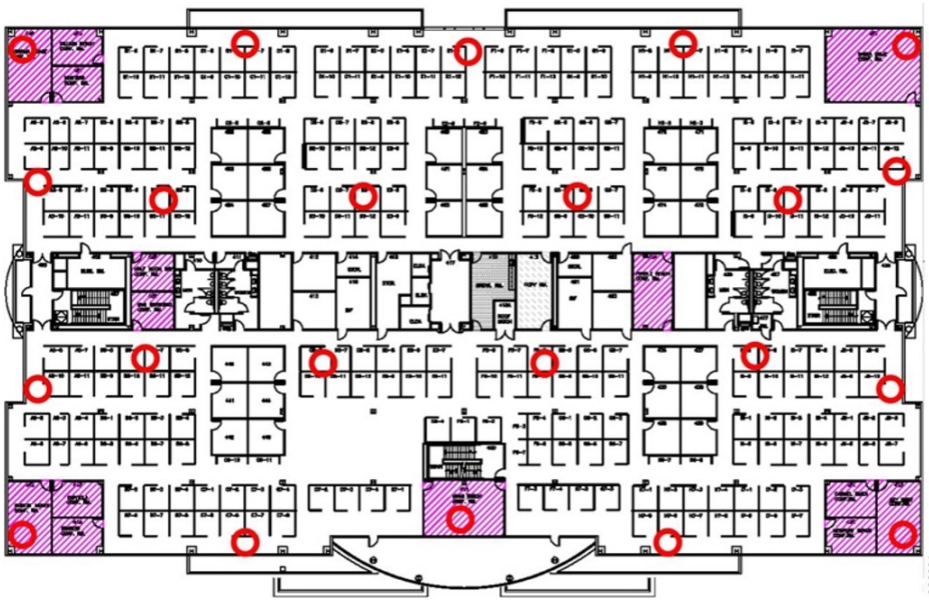

Figure 4.3: Wireless Access Point through the datacentre floor [5]

4.3 Current State of Art

Public Cloud Providers such as Google, Facebook, Microsoft Azure and Amazon AWS are governed by the GDPR regulations. Unlike the enterprise datacenters, the advantages these contemporary systems are a) all their infrastructure and data workflow are homogenous and b) non encumbered with the backward compatibility of applications and data sets to legacy apps.

We studied the Google infrastructure, Bigtable [79] and examined how it could be adopted to designing a system architecture which provides user the control of their data. The novelty of Bigtable is the simple data model, which provides application (and users) dynamic control over data layout and format, shown in Figure 4.4.

Bigtable is a distributed storage system for managing structured data that is designed to scale to a very large size: petabytes of data across thousands of commodity servers. The combination of Bigtable on GFS [80] has supported many projects at Google store data including web indexing, and Google Earth among others. These applications place very different demands on Bigtable, both in terms of data size (from URLs to web pages to satellite imagery) and latency requirements (from backend bulk

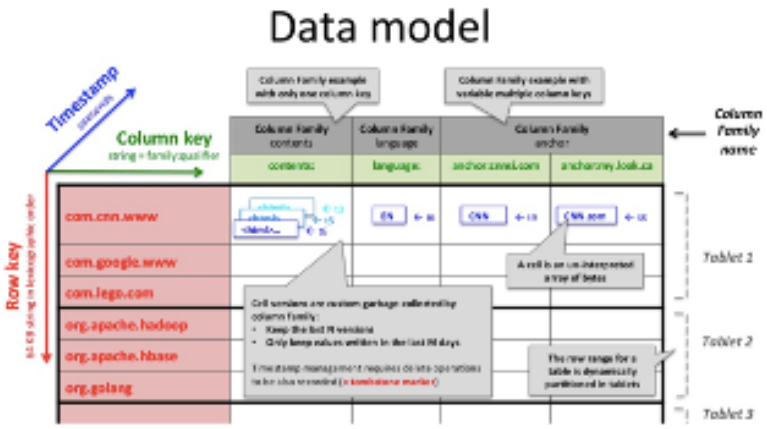

Figure 4.4: Data Model for Bigtable [6]

processing to real-time data serving). Despite these varied demands, Bigtable and GFS has successfully provided a flexible, high-performance solution for all of these Google products.

Theory of Operations: A transaction granularity is a row. A row is populated with the column family with arbitrary payload defined by the attributes in each column family. Every update is a new time series of row transaction. The most recent write is added to the data model in chunks of 64KB and can be read instantly. This provides high sustained bandwidth and low latency.

Key architectural assumptions: Bigtable and GFS implement well-defined semantics for concurrent appends [81, 82, 83]. This supports a strong set of producer-consumer model, hundreds of producers concurrently appending, provide atomicity with minimal synchronization overhead and consumers able to read the file as it is written. The control of data management is with client.

Inherent data management attribute provided: The application is always consistent and inherently has access to the versioned data, providing data protection as key inherent attribute. The column family can be easily extended with any new form of metadata at any given point of time without need for re-indexing or any disruptive set of operations to the data layout.

Key difference against relational databases and filesystems [84, 85, 86, 87]: The data model is very simple and scalable but in order to achieve the scale there are specific tradeoffs and differences between Bigtable vs Relationship databases. Some of these key differences are a) No data independence: Clients dynamically control locality by key names and whether to serve data from memory or disk, b) No multi-row transactions are supported in Bigtable, c) No table-wide integrity constraints, d) No Uninterpreted values, clients can marshal (serialize and de-serialize) data and e) No complex queries support, no SQL, all APIs are C++ requesting rows of data across the column family.

Similarly, there are fundamental difference with Google File System (GFS) and traditional filesystem. GFS is Not POSIX and doesn't guarantee any backward compatibility for POSIX compliant applications. Although GFS is not POSIX, through APIs for applications such as Bigtable Google supports operations like create, delete, open, close, read and write. GFS also supports snapshot and record append operations. GFS does not support random read/write within the file chunks, there is an implicit assumption in the data model that all the read/write are sequential in chunks of 64KB.

4.4 Proposal for Open Systems

The primary outcome of the research is to provide a framework to modernize all the Storage protocols in the hybrid cloud environment to factor in the security, regulatory and policy concerns.

The state of art Google infrastructure assumes new application model wherein backward compatibility for the legacy application is not required. This relaxed constraint is foundation for the scalable architecture for the Google infrastructure. It is easy to add metadata such as regulatory compliance identifiers for the data access as one of the column family in the Bigtable. Similarly, embedding GPS co-ordinates to be a foundation for enforcing data management could also be asynchronous addition to the column family in Bigtable. And similarly, secure connection management to avoid man in the middle attack can be enforced via Chubby and per-object based multi-factor authentication schemes.

All the three storage protocols, iSCSI, NFS and S3 are client server architecture with some changes around the way the protocols authenticate.

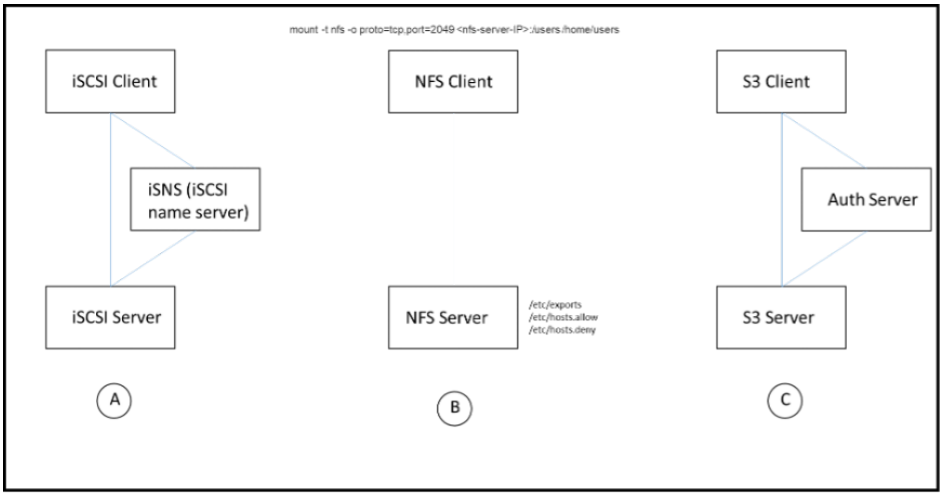

Figure 4.5: Client Server Protocols Theory of Operations

As shown in Figure 4.5, a) in case of iSCSI, the list of iSCSI servers can be optionally advertised to the clients through the iSCSI name server, b) the NFS server have entries in well placed directories /etc/ the file directory structure to export to various NFS clients allowed by the simple ipaddress entries in the /etc/hosts.allow and /etc/hosts.deny file, and c) the S3 Server authenticates the client with simple authentication server.

The three research topics will be focused on specifically answering the following questions, a) How can we embed compliance related metadata seamlessly in the access methods of traditional POSIX compliant applications, b) How can we build a model where the GPS co-ordinates are embedded in the metadata for POSIX compliant applications and c) Using iSCSI protocol as an example we will provide a mechanism to minimize the man in middle attack for the iSCSI connection in the hybrid cloud environment.

Embedding scalable access metadata in the data structures

The process flow for data access can be modelled with the following flow chart shown in Figure 4.6which embodies the different constraints and requirement highlighted in Table 4.2 in above sections.

Our proposal is we create a pluggable authentication module, called privacy access module which mandates the use of the library with the flow chart in Figure 4.6 prior

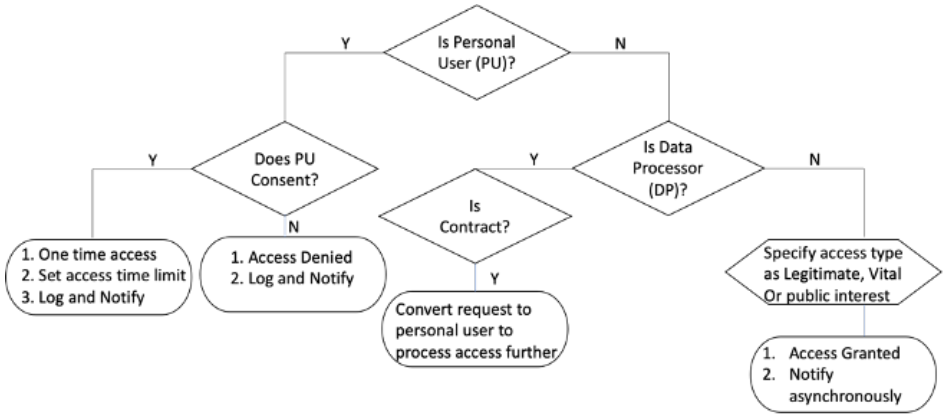

Figure 4.6: Flowchart for Privacy Module

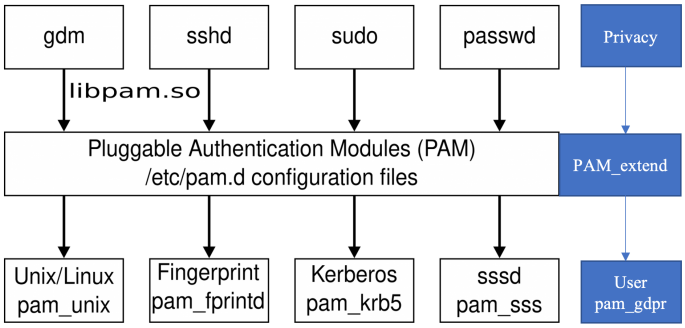

Figure 4.7: Linux Pluggable Authentical Module extended with PAM for privacy module

to any access to the data. In such a model, privacy of the personal user is centre of all the access pattern. The user is always notified of the access, either synchronously if the requestor is personal or the data processor and asynchronously post-event because the access was granted in case of the legal processor. This model provides the transparency GDPR is seeking for while making concessions for the legitimate access in cases such access is warranted.

The above Figure 4.7 provides the Linux privacy module [88], we added a privacy module which the metadata and the authentication which encompasses the GDPR primary intent to protect the privacy for the user data while also allowing for the access for legal requirement. This model is extensible with any of the new changes which might be incorporated and also extends the model which most of the applications are already

Table 4.4: Valid Opcodes for message sent by initiator to target

Opcode	Message encapsulated by the header
0x00	NOP-Out Message (from initiator to target)
0x01	SCSI Command (encapsulates a SCSI Command Descriptor Block)
0x02	SCSI Task Management Command
0x03	Login Command
0x04	Text Command
0x05	SCSI Data (for WRITE operation)
0x09	Ping Command (from initiator to target)
0x0a	Map Command

familiar with. The implementation can also be open sourced so that the transparency and security is protected.

4.4.1 Integrating Location Based Services in the existing protocols. Is it better done in-band to the storage control path or out of band?

There are performance and portability concerns in addressing this question. In-band control path implies an addition of new commands in existing protocols, ex: Table 4.4 and Table 4.5 are the Opcode (Operational Codes) for the iSCSI commands.

We can add another command x0b Location Command to the current list below, with corresponding response of 0x4b which provides the latitude, longitude information across geos and rack/row level info within the datacentre.

Out-band command is a separate machine which periodically checks all the initiators and targets along with the Apps and creates a topology of access pattern and based on policies monitors and enforces appropriate connections. The out of bound framework can be re-used for the NFS (file) and S3 (object) as well. Whereas the in-band command will be specific extension to the existing protocols.

Similarly, both NFS and the Object Protocols we can add the in-band commands. In case of NFS, since it is RPC based command sets between client and server, we can add a new RPC_LOC which will get and set the attribute related to the location. And in case of the object storage, we can add http function for locate ().

In this section we will examine various algorithms we can use to embed location info

Table 4.5: Valid Opcodes for responses (sent by target to initiator)

Opcode	Message encapsulated by the header
0x40	NOP-Out Message (from target to initiator)
0x41	SCSI Command (contains SCSI Status and Sense Information/Other response info)
0x42	SCSI Task Management Response
0x43	Login Response
0x44	Text Response
0x45	SCSI Data (for READ operation)
0x46	Ready to Transfer (from target to initiator when ready to receive data)
0x47	Asynchronous Event (from target to initiator indicating special conditions)
0x48	Opcode Not Understood
0x49	Ping Response (from target to initiator)
0x4a	Map Response

in Block (iSCSI), File (NFS) and Object (S3) protocols. Against these three protocols we will then examine the effect of latency/performance impact for both in band and out of band operations. The advantages of embedding the IO in the Storage protocol is we can assign policies to identify compliance for the data governance, identify malicious attack and also provide a good audit and tracking mechanisms for the mobile workloads. For all practical purposes, we will find examine simple server-client model for the three protocols with both the in-band and out of band options for embedding location-based information.

Approaches:

- a) Trusted Client In Band (TCB)
- b) Untrusted Client Out of Band (UCB)
- c) Untrusted Location Service Out of Band (ULB)

Approach A: Trusted Client In-band

The flow for the approach is Figure 4.8 depends on the Client been responsible for the location services embedding. The client stores the location information in its cache and then at every IO this location information is populated in the IO stream inline. Since

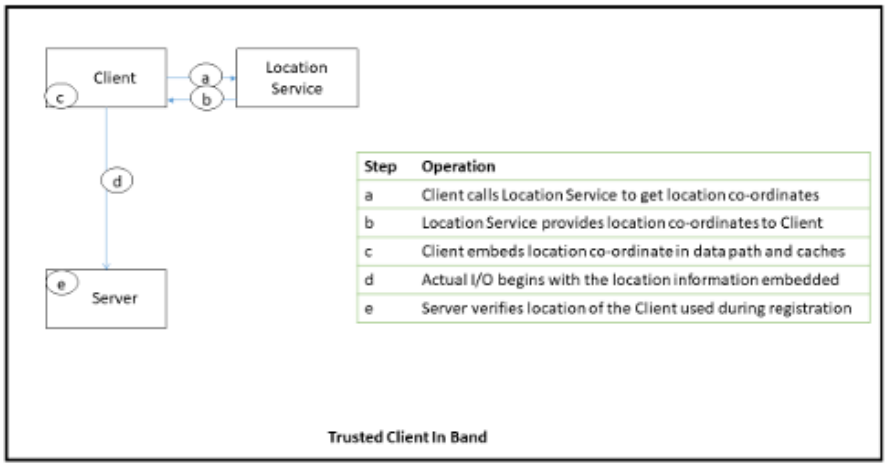

Figure 4.8: Flow for the trusted client in band

Server already has the client registration info, the client location at the registration is verified at every IO.

When during virtual machine motion, the Client is moved as virtue of the motion operation the registration info as well as the co-ordinates are updated for any new IO. The system can keep the records for all the locations it has been which can prove helpful for the provenance of the virtual machine and its associated data.

This approach is likely going to increase the latency of every IO in the system. Applications which are already sensitive to the latency may not be amiable to the inline overhead of location info passed through the system, hence there are other approaches which we will consider as well.

Approach B: Trusted Client In-band

In this approach, the burden of providing location to the IO is with the server. Post registration of the Client to the Server, when a write request is made the location info is embedded in the IO pattern, shown in Figure 4.9.

Because the location information is embedded in the IO data structure by every write to the system, any read request which doesn't below to the client can be easily identified by the system. The server been responsible for identifying the client locations and validating puts an overhead on the server, but this approach has an advantage of

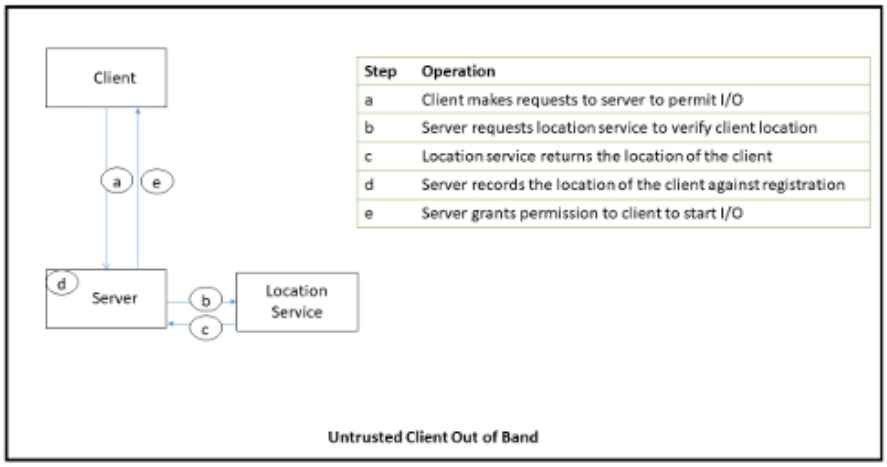

Figure 4.9: Flow for the untrusted client out of band

centralized system management and housekeeping.

Approach C: Untrusted Location Service out of Band

This is the most balanced of the approach in terms of responsibility. The responsibility of verifying the location of the IO is jointly that of both the server and the client in the storage protocol, shown in Figure 4.10.

Also, another advantage of this approach is that the location information is not inline embedded in the IO pattern of the storage protocol. The side band protocol has likely the performance and also data portability advantages as it is not polluting the existing storage protocols with new metadata information. In the next section we will examine the latency effects on the IO of the three approaches discussed in this section.

Preliminary Experiments and Results This research is opening new opportunities of research for combining the field on location-based services to the storage protocols. We used Systor 17 traces wherever possible to help with capturing the relative performance of the systems but most of the test were performed using test suites from Storage labs at Hewlett Packard Enterprise.

In the first part of this section, we will compare of the impact of the latency of the IO across the three protocols while including the storage protocols per the three approaches mentioned in Section 4 above.

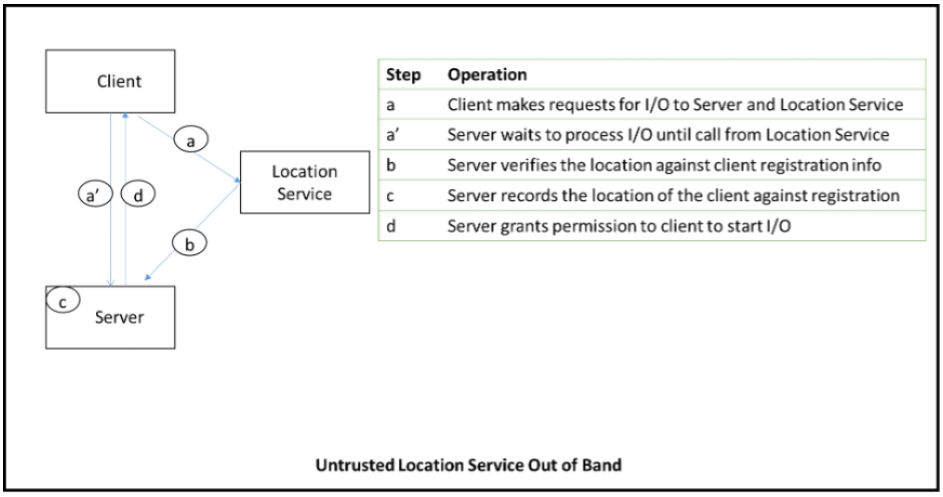

Figure 4.10: Flow for the untrusted location service out of band

Figure 4.11 shows that we can conclude that the in-band embedded approach Block Storage, iSCSI storage protocol the most. The deviation is 16

The latency impact on the block storage goes down in the embedded inline model as block sizes increases because more time is spent by the IO delivering the larger block payload while in parallel during the payload delivery the location information is collected, embedded by the client.

The out-band location-based service embed is largely immune to the size of the IO payload since almost all the work is done outside of the in-band data path.

From the Figure 4.12, NFS has the similar pattern of impact across the three different approaches of the location-based services integration in comparison with iSCSI. The biggest difference is in that the amplitude of the latency delta is relatively smaller.

This can be explained in that block storage has less overhead that the file storage and particularly in case of NFS which has the overhead of the full TCP and the file system and virtual file system stack which it needs to traverse. So essentially because the base latency for NFS is typically higher than the iSCSI the delta impact of the latency drop is not seen as much in the NFS protocol even for the embedded location service information embedded system.

In Figure 4.13, the S3 impact on various block sizes of the IO with the three different approaches are compared. The observations don't follow the pattern we observed in the

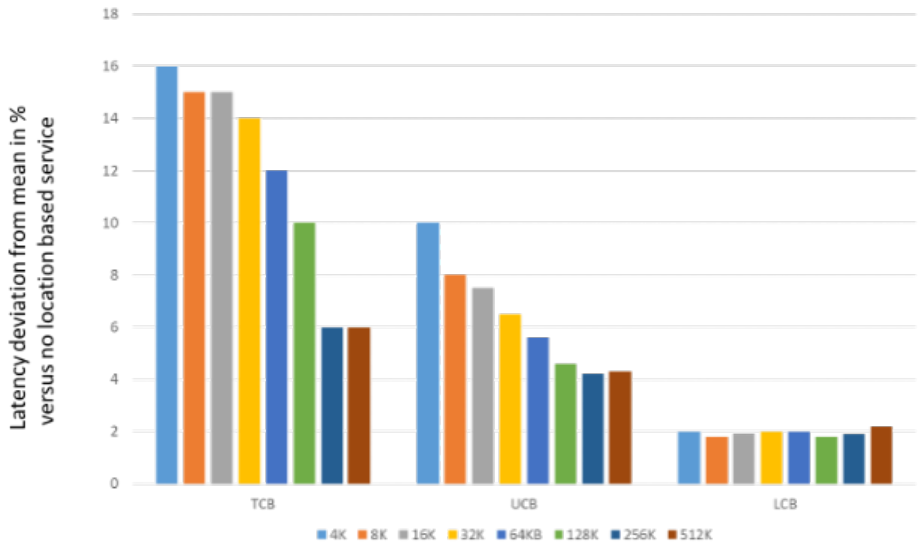

Figure 4.11: iSCSI Block Size v/s Impact of the embedded location services on various approaches

earlier two cases with iSCSI and NFS. The embedded option is very favorable compared to the out of bound approach. The reasons for this are S3 is a high latency over the internet protocols, the protocols are fundamentally designed to have meta data reach http interface, so embedding the location information in the data path inline is just like adding a custom attribute like the meta data search to the object. Additionally, we observe that the out of band protocols adds more latency to the location-based service, this can be explained by examining the protocols further. The cost of updating the metadata after the IO is higher since the location info is additional new IO over the internet latency. Our conclusion from the observation in Figure 4.11, 4.12, and 4.13 is that given how different the core latency principles for these three storage protocols, there is no one size fits all and our recommendations would be, a) for iSCSI, use out of band location-based services, b) for NFS, use untrusted client in-band and c) for S3, use in band per IO location-based services.

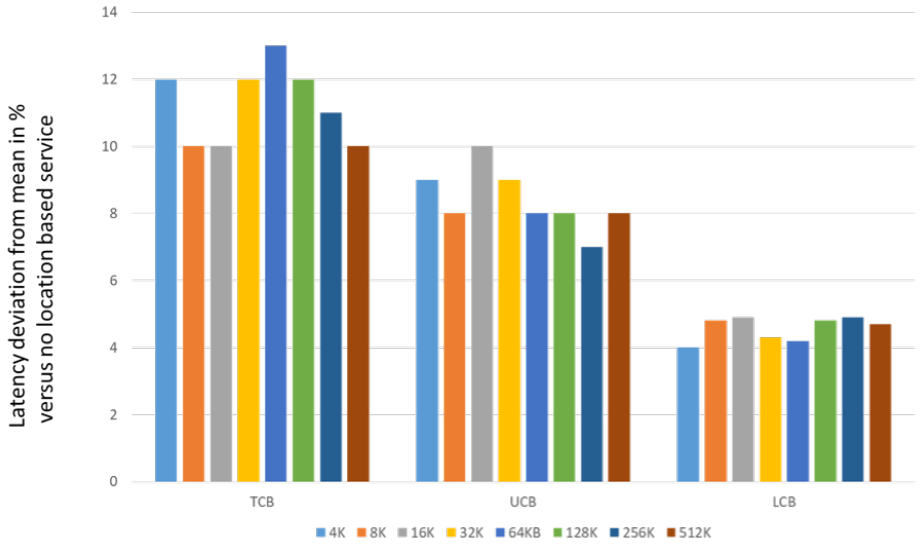

Figure 4.12: NFS Block Size v/s Impact of the embedded location services on various approaches

4.5 How do we avoid man in the middle manipulation of the location information?

We will examine different schemes to identify and rectify any possible manipulation via man in the middle attacks. We can envision scenarios where when a virtual machine or a container move either across clouds or within datacentre resulting an opportunity to introduce a wave of updates of block, file or object storage updates with the information about the new location. This might present a new denial of service opportunity. We plan to examine ways to detect and prevent such scenarios.

Temper Evident Registration

In this section we will propose an approach to develop a temper evident registration scheme [89, 90] for registration of the iSCSI Client to the iSCSI Server. This "registration technique" can minimize the man in the middle attack. The approach can be used for the other protocols such as NFS (file) and S3 (object). As shown in the Figure 4.14 below, the iSCSI client can be connected to the iSCSI server with authentication. The theory of operations is that any iSCSI client which wishes to connect to iSCSI Server in the Domain request a list of available servers in from the iSNS (iSCSI Name Service).

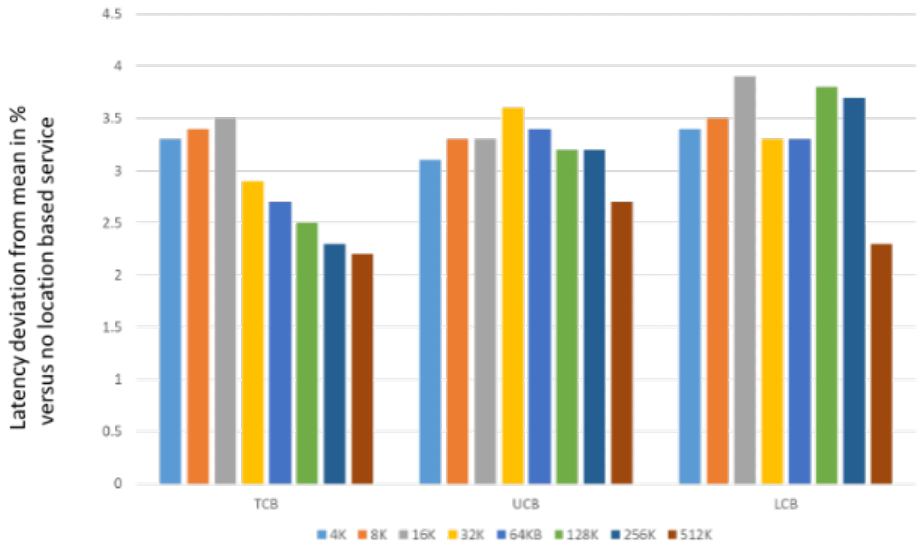

Figure 4.13: S3 Block Size v/s Impact of the embedded location services on various approaches

Now it is possible with a simple ARP poisoning attack for either a) fake iSNS Server or the b) fake iSCSI Client gain access to the important traffic in the iSCSI Network.

Inspired by the research work in the wireless pairing community by [80] we have developed a proposal for the temper evident registration scheme for the iSCSI registration.

The key attribute of the adoption of the TEP for wireless to the Storage protocol has following attributes. A) Time base limit for a registration session (default and typical 60 secs), B) Location Based Services and C) Synchronized NTP between the Server, Clients and the WAP (Wireless Access Point) or Location Based Services (LBS), shown in Figure 4.15.

Assumptions:

- All Clients, Servers, Location Based Services and iSNS Servers are time synchronized with NTP
- Registration process is set as 60 seconds. If a registration is not complete in 60 seconds, retry is possible. Only two retries are allowed. If both re-tries fail, the IP address and MAC address is back listed by the server and manual intervention

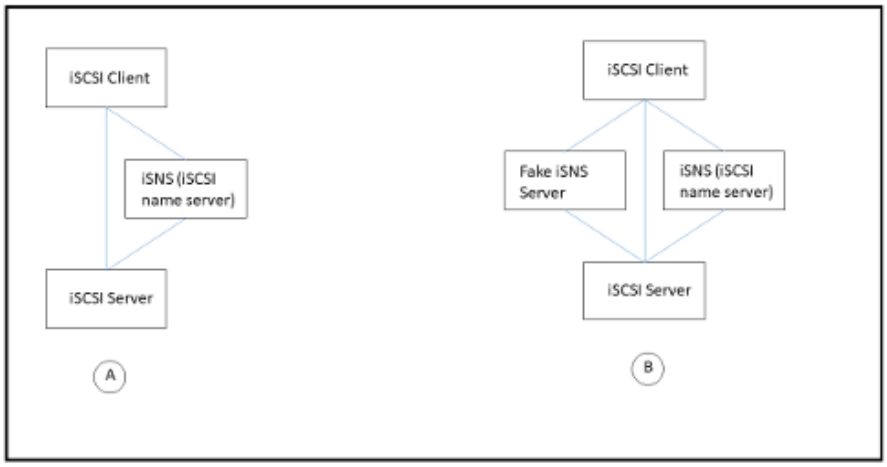

Figure 4.14: Workflow between the iSCSI Server-Client and APR poisoning attack

is needed by the sys admin to administer the blacklisted IP address and the MAC address.

- The Clients have all the software and hardware components to respond to either GPS based or the Wireless Location Based Services

Steps of registration in the steady state:

- iSCSI client requests the name of all the iSCSI Servers in the Domain. This is requested from the iSNS (iSCSI Name Server). The iSCSI name server returns the list of the iSCSI server in the Domain.
- Once the iSCSI client receives the name of the iSCSI servers, it makes a request for connection to the iSCSI server. When such a request is made, the iSCSI server creates a timestamp, say t1 against the ipaddress and MAC address (virtual and physical) of the request. The registration window of 60 sec now begins. iSCSI server responds request acknowledgement along with the time of initial registration to the client.
- The iSCSI server requests to the local location-based server to verify the location of the client. The location-based server triangulates the location of the server in

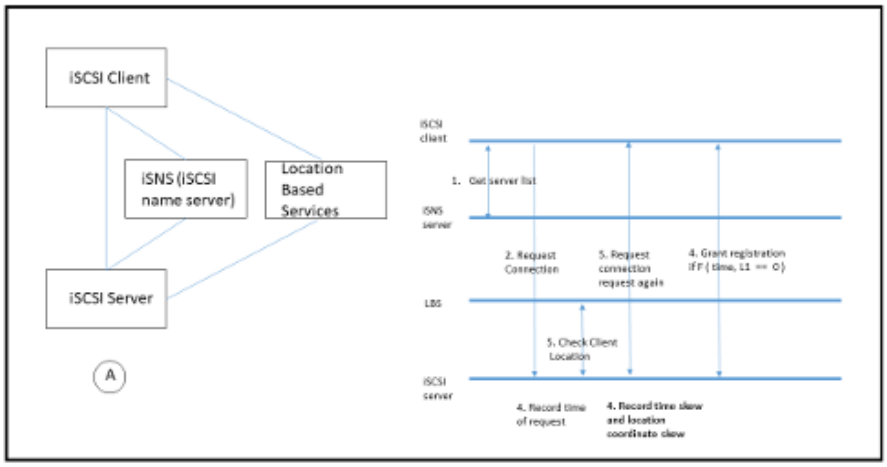

Figure 4.15: Flow of connection requests for registration between client-server in steady state

the datacenter and returns the address in the form of latitude, longitude, rack number, row number to the Server.

- The iSCSI server verifies if the location is in the valid location list, i.e., in policy for the data access. Once it does, the server sends a callback request to the client asking for interest in connection. As part of the response from the client to the server, the client sends the offset of the time left to complete the registration.
- If the offset of the time client has matches with the offset of time left on the server for registration, the server grants a session connection to the client.

Steps of registration in the state where fake iSNS Server is actively present in the domain as shown in Figure 4.16.

- iSCSI client requests the name of all the iSCSI Servers in the Domain. This is requested from the iSNS (iSCSI Name Server). The iSCSI name server returns the list of the iSCSI server in the Domain. However, the request was made to the Fake iSNS server.
- The fake iSNS server makes a request to the iSCSI server slightly ahead in time of the actual request coming to the iSCSI server. In other words, the real iSCSI

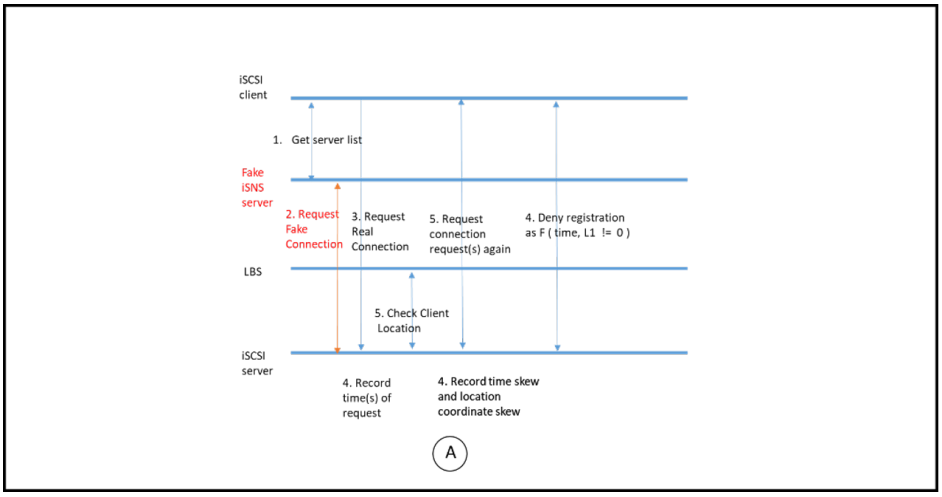

Figure 4.16: Flow of connection requests for client-server registration with fake iSNS Server

request from the client to the server is lagged by the fake iSNS server request for the client connection. 3. The iSCSI Server now receives two request(s), one from the fake iSNS server and second

- The iSCSI Server now receives two request(s), one from the fake iSNS server and second from the real iSCSI client. The iSCSI server at this point can't distinguish between the real or fake connections just yet. As per the protocol when the connection was initiated by the Server, the 60 sec time-window was started for both the connection requests.

- The Server makes location requests for the two connections to which it receives two different locations as expected (since the location of the real client and iSNS server are different).

- When the Server makes a re-establish connection request to the two connections, the real client (2nd inbound client to server connect) correctly responds with the proper time offset remaining for registration and the location. Whereas the Server detects tamper evident on the first connection because the time offset won't match a fake iSCSI client, or its location information will be skewed and the 1st inbound connection for authentication will be declined as temper evident and the ipaddress

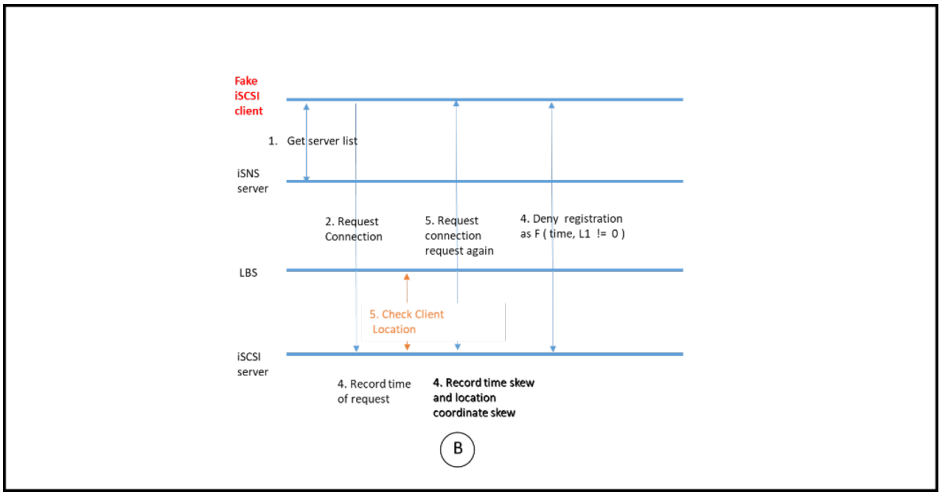

Figure 4.17: Flow of connection requests for client-server registration with fake iSCSI Client

and MAC address will be blacklisted.

Steps of registration in the state where fake iSNS Server is actively present in the domain as shown in Figure 4.17

- iSCSI client requests the name of all the iSCSI Servers in the Domain. This is requested from the iSNS (iSCSI Name Server). The iSCSI name server returns the list of the iSCSI server in the Domain. However, the request was made by the Fake iSCSI client.
- The fake iSCSI client makes a request to the iSCSI server based on the information it received from the iSNS server.
- The iSCSI Server now receives request from the fake iSCSI client, the iSCSI server at this point can't distinguish between the real or fake connections just yet. As per the protocol when the connection was initiated by the Server, the 60 sec time-window will be started for the connection requests.
- The Server makes location requests for the iSCSI client request it received. If the iSCSI client ipaddress is legitimate in the Domain, the iSCSI server will receive two location request inquiry call-back one from the real client and other from the

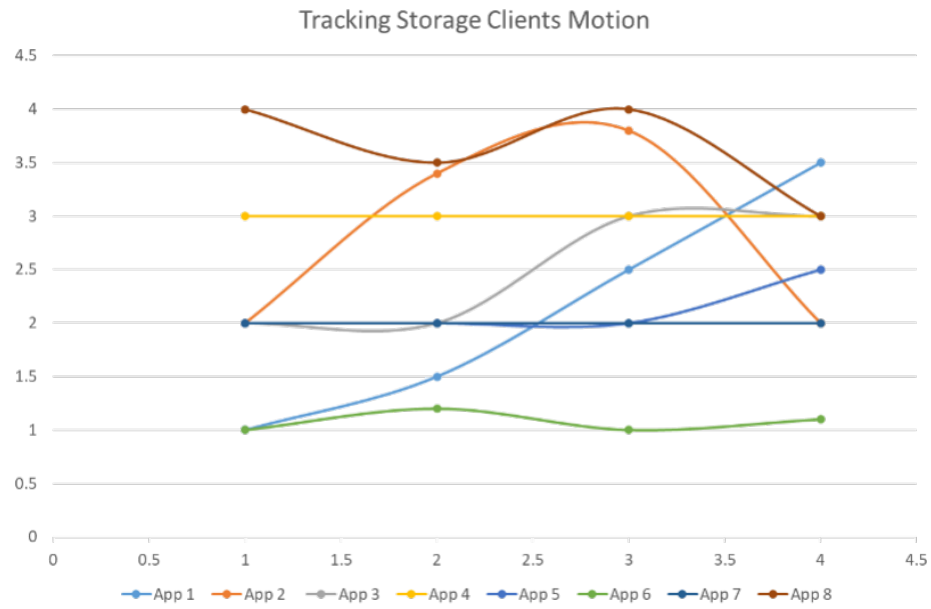

Figure 4.18: Storage Protocol Client motions in the rack design center

fake client, thereby the iSCSI server knows the connection registration is tempered and hence request denied.

- Furthermore, iSCSI server can recognize the fake iSCSI client because the registration time skew (offset) will not make for the fake iSCSI client. The iSCSI server could allow the second connection to go through but, in our implementation, we blocked both the connection to the same ipaddress.

In Figure 4.18, we observe the various applications movement in the rack design center based on the virtual machine motion over time in the data center. As is observed because of the granular location-based services applied, we can track the actual location and the movement of the data store for the virtual machines throughout the lifecycle of the data. This visualization is very useful visualization to track the lifecycle of virtual machine.

The above visualization provides a very easy way to enforce location-based policy for compliance and regulatory reasons. Figure 4.19 provides a multi-app to the rack and row mapping for the virtual machines which reside in the infrastructure.

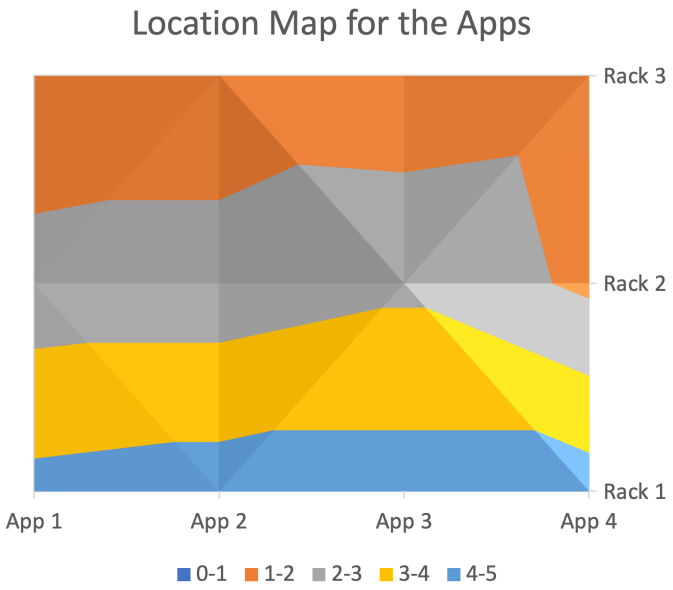

Figure 4.19: Provides view of the Apps assigned against the rows/columns of racks

A snapshot of such much can be periodically taken by either the sys admin or the location services to monitor the mobility of the virtual workload.

Trying to find an optimum registration window for the servers to connect with the client is desired. Too much time and there can be more MTM denial of service attacks and too little time window and the registration process will be flooded with a lot of false connection drops.

We simulated a few MTM attacks across the three popular storage protocols. Our tests were done with the 60 sec default time windows based on the observation above, we believe we can reduce the registration window safely to 30 seconds, shown in Figure 4.20.

4.6 Related Work

Provenance for cloud storage was an active research area in mid last couple of decades [91, 92, 93, 94, 95, 96]. Provenance is metadata that describes the history of the object. Most of the research area was focused around making the object or data very self-aware of its entire history. This research didn't have the benefit of advances in the location

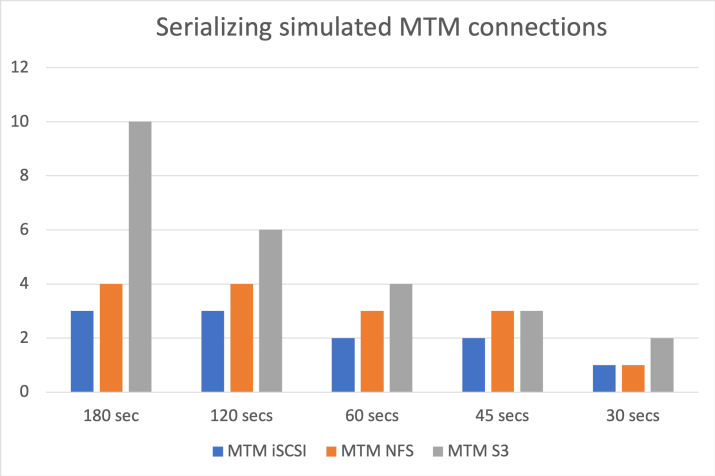

Figure 4.20: Provides simulated experiment to handle the MTM attacks in different windows

aware, both GPS and also the Wireless Access Point technologies. Also, the research was focused on introducing this concept to the operating system level which make it hard to implement and adopt. The amount of data needed to keep the provenance of all the objects in the systems was very large. Our research studies the various approach taking in to account the best place to introduce a new attribute in the infrastructure and its impact on scale and performance of the system.

Much of the research in the Storage Protocols have been focused on securing the end-to-end data encryption and improving the latency of the Protocols themselves. The Storage industry had new media types emerge such as Flash, Non-Volatile Memory which has made much of the industry research focus on the effective use of the underlying media changes. Also, data integrity and encryption, Kerberos integration's for authentication are also relatively new research areas in this Storage industry. Our research of the storage industry has informed us to make sure whichever implementation path we take; portability of applications is a key attribute we want to retain as much as possible. New applications can incrementally take advantage of the new features, but it is incredibly hard to adopt new features if it fundamentally modifies how Storage is accessed by the applications.

Location Based Services has been studied and researched extensively in the mobile applications. The advantage this domain had over the hybrid cloud is that the

applications were mostly green field. There are a number of lessons learnt from the experience in the mobile applications which we plan to leverage for implementation and scaling location-based services in the Storage Protocols for end-to-end visibility from applications to storage.

4.7 Future Work

Location Based Services in Storage Protocol is a brand-new field with a lot of research potential. Some of the future research areas we envision are a) adding more than one location-based services to the study to prevent the location-based services themselves is not comprised, b) integrate location based with Kerberos and other authentication services already used by the storage protocols, c) using caching, look-ahead techniques to improve performance while in embedded in-band mode of including the location-based services in the protocols.

Ability to detect real-time security anomalies: Can an end-to-end location aware infrastructure also determine an unintended virtual machine/container assessing the data stores? Anomalies such as high read activity, irregular access pattern could be a signature for malicious activities on the data set. Applying various machine learning techniques identify any malicious virtual machines or containers in the environment. Such anomalies could be used to detect security issues as well as help point to an effective and economic way to use hybrid cloud. We envision that we will have to explore use of machine learning techniques to handle the wide amount of metadata transformation while testing at scale the proposed framework.

4.8 Conclusion

Building Regulatory Compliant Hybrid Cloud is aspiration whether be it public cloud or the private cloud enterprises or both. However, Storage Protocols have not fundamentally changed since their emergence from the SCSI era to NFS era and now the object storage era. The big challenge is needed to maintain portability of applications while making them secure through location-based services. The novelty of the research is in the ability to balance the portability with the practicality for the inclusion of

location-based services in the storage protocols.

We believe this research will ignite and start various fundamental research in making storage protocols modern, scalable and mobile in the cloud first world. The framework proposed will be foundation for an entire new field of application of machine learning to solving the scale problems in the Storage domain introduced and applied in this research.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

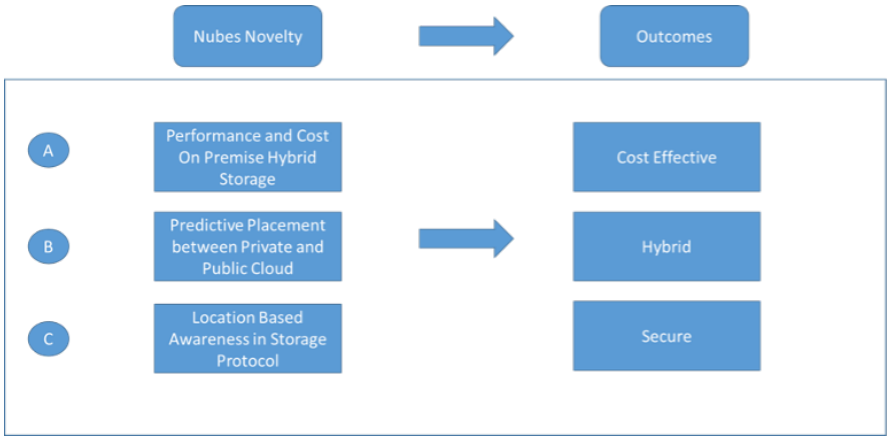

Figure 5.1: Snapshot of the Nubes framework Novelty and the Outcomes

The thesis focused on the need to build the secure, scalable and cost effective hybrid cloud. The figure 5.1 sums up the three key areas which we focused our research on with the Nubes framework. In the Chapter 2, focus of the research was to make sure that with the advent of the new media technologies like Flash we find an effective way to deliver the best performance in a constraint resource envelope a hybrid storage system which uses modern machine learning techniques like support vector machine to find the best way to deliver the best price performance from the hybrid array in a private

cloud. In the Chapter 3, focus of the research was to predict whether a new workload should be placed in the public or private cloud. The technique of pre-determination can also be applied to existing workload should its characteristics change. Given data has gravity and it is impractical to move a large quantities of data have a pre-determined machine learning based algorithm to help predict where to place the workload is very important in the hybrid cloud environment. In the Chapter 4, focus on the research was to introduce a ground breaking concept of adding location based services to the storage protocols to minimize the man in the middle attack during client server connection registration and also embedding location based services in the storage protocol. The techniques described in the chapter can help fuel more research in the area of wide scale data management.

5.2 Future Work

The thesis is basis for fueling three major areas of future research work. First, solving issues in building large scale mixed storage media and system architecture private cloud systems. Second, solving issues in area of high performance and cost effective workload management in Hybrid Cloud environment. Third, solving issues in area of data privacy and security while allowing flexibility for the optimum workload and data management.

- Hybrid Systems: Three are three trends influencing the hybrid storage system architectures. First, a spectrum of advances in media, on one end high density storage to improve cost per storage unit and on the other end low latency media for high performance. Second, a single storage domain name-space which spans from Internet of Things, Private Cloud(s) to Public Cloud(s). Third, large working set in TB's and PB's which spans any single domain of architecture. The thesis focused on two-class hybrid storage systems. Further research is needed in multi-class hybrid storage systems. Concepts of caching, data tiering and data staging need further research in such multi-class hybrid storage systems. Data placement algorithms which make an efficient trade-off between cost and performance in determination of caching, tiering and staging on specific data sets in large multi-class hybrid storage systems are intriguing future research areas.

- High Performance and Cost Effective Workload Management: More and more applications are containerized. This trend is essentially creating a clean and clear separation of the business logic of application from the destination of the data sets. The data sets can truly be in any of clouds. Between virtual machines and the containers the workloads are highly virtualized and runs anywhere. More research is needed in the virtualization of the data sets so not only the optimal data placement can be made but also pre-fetch and optimal initial and run time workload movement across hybrid clouds can be determined.

- Data Privacy and Security: This is a relatively new field of research. In this thesis we focused on use of location based services in the low latency block storage portfolio. The use of location based services to determine batch operations for analytic, intrusion detection in real (or run) time of the workload deployment (migration), providing clear security models where the end user as well as the service providers can easily determine if there are breaches or non-conformance of compliance regulations are new areas of research. Further, there are research opportunities to extend the POSIX standards to modernize the protocols or define new standards to advance the open source communities to make data privacy and security inherent in the storage and data protocols.

Hybrid Cloud architecture are still in early stages of deployments and as much and more applications truly can be run and deployed regardless of cloud destination, the three areas of hybrid storage systems, high performance and cost effective workload management and data privacy, security inherent in system architectures have large amount of fundamental research in algorithm for system architecture at scale yet to be researched.

References

[1] Amazon web service. https://aws.amazon.com/.

[2] Oracle paas and iaas public cloud services . https://www.oracle.com/assets/paas-iaas-pub-cld-srvs-pillar-4021422.pdf.

[3] Microsoft azure. https://azure.microsoft.com/en-us/.

[4] Google cloud. https://cloud.google.com/.

[5] Understanding google bigtable non-sql attributes. https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180AB375.

[6] France Romain Jacotin, Orange Telecom. An example of an indoor (data center) wireless access point location based on cisco. https://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Mobility/WiFiLBS-DG/wifich5.html.

[7] Storage performance council(university of massa-chusetts trace repository). http://traces.cs.umass.edu/index.php/Storage, 2007.

[8] Fenggang Wu, Bingzhe Li, Zhichao Cao, Baoquan Zhang, Ming-Hong Yang, Hao Wen, and David HC Du. Zonealloy: Elastic data and space management for hybrid {SMR} drives. In *11th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 19)*, 2019.

[9] Kan Wu, Andrea Arpaci-Dusseau, and Remzi Arpaci-Dusseau. Towards an unwritten contract of intel optane ssd. In *11th USENIX Workshop on Hot Topics*

in Storage and File Systems (HotStorage 19). USENIX Association, Renton, WA, 2019.

[10] Snia solid state storage: The key to the next gen solid state storage technologies. `http://www.snia.org/sites/default/files/AnilVasudeva_Solid_State_Storage_Key_NextGen.pdf`.

[11] Zhengyu Yang, Morteza Hoseinzadeh, Allen Andrews, Clay Mayers, David Thomas Evans, Rory Thomas Bolt, Janki Bhimani, Ningfang Mi, and Steven Swanson. Autotiering: automatic data placement manager in multi-tier all-flash datacenter. In *Performance Computing and Communications Conference (IPCCC), 2017 IEEE 36th International*, pages 1–8. IEEE, 2017.

[12] Ilias Iliadis, Jens Jelitto, Yusik Kim, Slavisa Sarafijanovic, and Vinodh Venkatesan. Exaplan: Efficient queueing-based data placement, provisioning, and load balancing for large tiered storage systems. *ACM Transactions on Storage (TOS)*, 13(2):17, 2017.

[13] Jorge Guerra, Himabindu Pucha, Joseph S Glider, Wendy Belluomini, and Raju Rangaswami. Cost effective storage using extent based dynamic tiering. In *FAST*, volume 11, pages 20–20, 2011.

[14] Haixiang Shi, Rajesh Vellore Arumugam, Chuan Heng Foh, and Kyawt Kyawt Khaing. Optimal disk storage allocation for multi-tier storage system. In *APMRC, 2012 Digest*, pages 1–7. IEEE, 2012.

[15] Eric Anderson, Susan Spence, Ram Swaminathan, Mahesh Kallahalla, and Qian Wang. Quickly finding near-optimal storage designs. *ACM Transactions on Computer Systems (TOCS)*, 23(4):337–374, 2005.

[16] Bingzhe Li, Yaobin Qin, Bo Yuan, and David J Lilja. Neural network classifiers using stochastic computing with a hardware-oriented approximate activation function. In *2017 IEEE 35th International Conference on Computer Design (ICCD)*, pages 97–104. IEEE, 2017.

[17] Terrence S Furey, Nello Cristianini, Nigel Duffy, David W Bednarski, Michel Schummer, and David Haussler. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16(10):906–914, 2000.

[18] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[19] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification. 2003.

[20] Feng Chen, David A Koufaty, and Xiaodong Zhang. Hystor: making the best use of solid state drives in high performance storage systems. In *Proceedings of the international conference on Supercomputing*, pages 22–32. ACM, 2011.

[21] Mustafa Canim, George A Mihaila, Bishwaranjan Bhattacharjee, Kenneth A Ross, and Christian A Lang. Ssd bufferpool extensions for database systems. *Proceedings of the VLDB Endowment*, 3(1-2):1435–1446, 2010.

[22] Yanfei Lv, Bin Cui, Xuexuan Chen, and Jing Li. Hat: an efficient buffer management method for flash-based hybrid storage systems. *Frontiers of Computer Science*, 8(3):440–455, 2014.

[23] Xiongzi Ge, Xuchao Xie, David HC Du, Pradeep Ganesan, and Dennis Hahn. Chewanalyzer: Workload-aware data management across differentiated storage pools. In *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 94–101. IEEE, 2018.

[24] Ishwar Krishnan Sethi and GPR Sarvarayudu. Hierarchical classifier design using mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, (4):441–445, 1982.

[25] Dushyanth Narayanan, Austin Donnelly, and Antony Rowstron. Write off-loading: Practical power management for enterprise storage. *ACM Transactions on Storage (TOS)*, 4(3):10, 2008.

[26] Chunghan Lee, Tatsuo Kumano, Tatsuma Matsuki, Hiroshi Endo, Naoto Fukumoto, and Mariko Sugawara. Understanding storage traffic characteristics on enterprise virtual desktop infrastructure. In *Proceedings of the 10th ACM International Systems and Storage Conference*, page 13. ACM, 2017.

[27] Akshat Verma, Ricardo Koller, Luis Useche, and Raju Rangaswami. Srcmap: Energy proportional storage using dynamic consolidation. In *FAST*, volume 10, pages 267–280, 2010.

[28] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, 2010.

[29] Nimrod Megiddo and Dharmendra S Modha. Arc: A self-tuning, low overhead replacement cache. In *FAST*, volume 3, pages 115–130, 2003.

[30] Giuseppe Vietri, Liana V Rodriguez, Wendy A Martinez, Steven Lyons, Jason Liu, Raju Rangaswami, Ming Zhao, and Giri Narasimhan. Driving cache replacement with ml-based lecar. In *10th {USENIX} Workshop on Hot Topics in Storage and File Systems (HotStorage 18)*, 2018.

[31] Zhichao Li. *GreenDM: A versatile tiering hybrid drive for the trade-off evaluation of performance, energy, and endurance*. PhD thesis, The Graduate School, Stony Brook University: Stony Brook, NY., 2014.

[32] Youngjae Kim, Aayush Gupta, Bhuvan Urgaonkar, Piotr Berman, and Anand Sivasubramaniam. Hybridstore: A cost-efficient, high-performance storage system combining ssds and hdds. In *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2011 IEEE 19th International Symposium on*, pages 227–236. IEEE, 2011.

[33] John D Strunk. Hybrid aggregates: Combining ssds and hdds in a single storage pool. *ACM SIGOPS Operating Systems Review*, 46(3):50–56, 2012.

[34] Hyojun Kim, Sangeetha Seshadri, Clement L Dickey, and Lawrence Chiu. Evaluating phase change memory for enterprise storage systems: A study of caching and tiering approaches. *ACM Transactions on Storage (TOS)*, 10(4):15, 2014.

[35] A Elnably and P Varman. Application-sensitive qos scheduling in storage servers. In *ACM Symposium on Parallelism in Algorithms and Architecture*, 2012.

[36] Magnus Karlsson, Christos Karamanolis, and Xiaoyun Zhu. Triage: Performance differentiation for storage systems using adaptive control. *ACM Transactions on Storage (TOS)*, 1(4):457–480, 2005.

[37] Hui Wang and Peter J Varman. Balancing fairness and efficiency in tiered storage systems with bottleneck-aware allocation. In *FAST*, volume 14, pages 229–242, 2014.

[38] Ji Xue, Feng Yan, Alma Riska, and Evgenia Smirni. Storage workload isolation via tier warming: How models can help. In *ICAC*, pages 1–11, 2014.

[39] Bingzhe Li, Chunhua Deng, Jinfeng Yang, David Lilja, Bo Yuan, and David Du. Haml-ssd: A hardware accelerated hotness-aware machine learning based ssd management. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8. IEEE, 2019.

[40] Tracy F Sienknecht, Richard J Friedrich, Joseph J Martinka, and Peter M Friedenbach. The implications of distributed data in a commercial environment on the design of hierarchical storage management. *Performance Evaluation*, 20(1-3):3–25, 1994.

[41] Dingshan He, Xianbo Zhang, David HC Du, and Gary Grider. Coordinating parallel hierarchical storage management in object-base cluster file systems. In *Proceedings of the 23rd IEEE Conference on Mass Storage Systems and Technologies (MSST)*, 2006.

[42] Jason Lango. Toward software-defined slas. *Communications of the ACM*, 57(1):54–60, 2014.

[43] Rahul Ghosh, Giribabu V Paramkusham, Aaron J Quirk, and Upendra Sharma. Selecting virtual machines to be migrated to public cloud during cloud bursting based on resource usage and scaling policies, March 28 2017. US Patent 9,606,826.

[44] Moustafa Najm and Venkatesh Tamarapalli. Vm migration for profit maximization in federated cloud data centers. In *2020 International Conference on COMmunication Systems & NETworkS (COMSNETS)*, pages 882–884. IEEE, 2020.

[45] Zhe Liu, Changle Li, Weijie Wu, and Riheng Jia. A hierarchical approach for resource allocation in hybrid cloud environments. *Wireless Networks*, 24(5):1491–1508, 2018.

[46] Helan Liang, Yanhua Du, Enting Gao, and Jinghan Sun. Cost-driven scheduling of service processes in hybrid cloud with vm deployment and interval-based charging. *Future Generation Computer Systems*, 107:351–367, 2020.

[47] Srinivas Byatarayanapura Venkataswamy, Indrajit Mandal, and Seetharam Keshavarao. Chicwhale optimization algorithm for the vm migration in cloud computing platform. *system*, 14:15.

[48] Michael Nelson, Beng-Hong Lim, Greg Hutchins, et al. Fast transparent migration for virtual machines. In *USENIX Annual technical conference, general track*, pages 391–394, 2005.

[49] Michael R Hines, Umesh Deshpande, and Kartik Gopalan. Post-copy live migration of virtual machines. *ACM SIGOPS operating systems review*, 43(3):14–26, 2009.

[50] Takahiro Hirofuchi, Hidemoto Nakada, Satoshi Itoh, and Satoshi Sekiguchi. Enabling instantaneous relocation of virtual machines with a lightweight vmm extension. In *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 73–83. IEEE, 2010.

[51] Hai Jin, Li Deng, Song Wu, Xuanhua Shi, and Xiaodong Pan. Live virtual machine migration with adaptive, memory compression. In *2009 IEEE International Conference on Cluster Computing and Workshops*, pages 1–10. IEEE, 2009.

[52] Jihun Kim, Dongju Chae, Jangwoo Kim, and Jong Kim. Guide-copy: fast and silent migration of virtual machine for datacenters. In *SC'13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pages 1–12. IEEE, 2013.

[53] Fei Zhang, Guangming Liu, Xiaoming Fu, and Ramin Yahyapour. A survey on virtual machine migration: Challenges, techniques, and open issues. *IEEE Communications Surveys & Tutorials*, 20(2):1206–1243, 2018.

[54] Xiang Zhang, Zhigang Huo, Jie Ma, and Dan Meng. Exploiting data deduplication to accelerate live virtual machine migration. In *2010 IEEE international conference on cluster computing*, pages 88–96. IEEE, 2010.

[55] TianZhang He, Adel N Toosi, and Rajkumar Buyya. Performance evaluation of live virtual machine migration in sdn-enabled cloud data centers. *Journal of Parallel and Distributed Computing*, 131:55–68, 2019.

[56] George EP Box and David A Pierce. Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *Journal of the American statistical Association*, 65(332):1509–1526, 1970.

[57] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[58] Jyotiska Nath Khasnabish, Mohammad Firoj Mithani, and Shrisha Rao. Tier-centric resource allocation in multi-tier cloud systems. *IEEE Transactions on Cloud Computing*, 5(3):576–589, 2015.

[59] Takfarinas Saber, James Thorburn, Liam Murphy, and Anthony Ventresque. Vm reassignment in hybrid clouds for large decentralised companies: A multi-objective challenge. *Future Generation Computer Systems*, 79:751–764, 2018.

[60] Jinjin Wang, Yonglong Zhang, Junwu Zhu, and Yi Jiang. A double auction vm migration approach. In *2nd EAI International Conference on Robotic Sensor Networks*, pages 141–147. Springer, 2020.

[61] Bo An, Victor R Lesser, David E Irwin, and Michael Zink. Automated negotiation with decommitment for dynamic resource allocation in cloud computing. In *AAMAS*, volume 10, pages 981–988, 2010.

[62] Hadi Goudarzi and Massoud Pedram. Multi-dimensional sla-based resource allocation for multi-tier cloud computing systems. In *2011 IEEE 4th International Conference on Cloud Computing*, pages 324–331. IEEE, 2011.

[63] Getzi Jeba Leelipushpam Paulraj, Sharmila Anand John Francis, J Dinesh Peter, and Immanuel Johnraja Jebadurai. A combined forecast-based virtual machine migration in cloud data centers. *Computers & Electrical Engineering*, 69:287–300, 2018.

[64] Hua He, Yu Zhao, and Shanchen Pang. Stochastic modeling and performance analysis of energy-aware cloud data center based on dynamic scalable stochastic petri net. *Computing and Informatics*, 39(1-2):28–50, 2020.

[65] Gaochao Xu, Junjie Pang, and Xiaodong Fu. A load balancing model based on cloud partitioning for the public cloud. *Tsinghua Science and Technology*, 18(1):34–39, 2013.

[66] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. Live migration of virtual machines. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, pages 273–286, 2005.

[67] Nikos Tziritas, Thanasis Loukopoulos, Samee Khan, Cheng-Zhong Xu, and Albert Zomaya. Online live vm migration algorithms to minimize total migration time and downtime. In *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 406–417. IEEE, 2019.

[68] Naga Malleswari Tyj and G Vadivu. Adaptive deduplication of virtual machine images using akka stream to accelerate live migration process in cloud environment. *Journal of Cloud Computing*, 8(1):3, 2019.

[69] Gdpr regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation). `https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32016R0679`. 2016.

[70] California consumer privacy act (ccpa) is a state statute to enhance privacy rights. https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180AB375. 2016.

[71] Michal S Gal and Oshrit Aviv. The competitive effects of the gdpr. *Journal of Competition Law & Economics*, 16(3):349–391, 2020.

[72] Gdpr readiness checklist. https://www.synopsys.com/software-integrity/resources/white-papers/gdpr-checklist.html. 2016.

[73] Data governance and stewardship: Designing data stewardship entities and advancing data access. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2965885/.

[74] The principles and the best practices for data governance in hybrid cloud environment. https://services.google.com/fh/files/misc/principles_best_practices_for_data-governance.pdf.

[75] Platform for applying the geo tag open-source location services to various applications. http://geo2tag.org/index.php/Geo2tag:Open_Source_LBS_Platform.

[76] Henning Maass. Location-aware mobile applications based on directory services. *Mobile Networks and Applications*, 3(2):157–173, 1998.

[77] Android development. http://developer.android.com/google/play-services/location.html.

[78] Google location services. http://developer.android.com/google/play-services/location.html.

[79] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C Hsieh, Deborah A Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E Gruber. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, 26(2):1–26, 2008.

[80] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The google file system. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 29–43, 2003.

[81] Google bigtable introduction and performance analysis. `https://www.cs.rochester.edu/courses/261/spring2017/termpaper/16/paper.pdf`.

[82] James C Corbett, Jeffrey Dean, Michael Epstein, Andrew Fikes, Christopher Frost, Jeffrey John Furman, Sanjay Ghemawat, Andrey Gubarev, Christopher Heiser, Peter Hochschild, et al. Spanner: Google's globally distributed database. *ACM Transactions on Computer Systems (TOCS)*, 31(3):1–22, 2013.

[83] Google file system explained in a tutorial at stanford university. `https://cs.stanford.edu/~matei/courses/2015/6.S897/slides/gfs.pdf`.

[84] Differences between hbase and bigtable. `https://cloud.google.com/bigtable/docs/hbase-differences`. 2016.

[85] Gheorghe Matei and Romanian Commercial Bank. Column-oriented databases, an alternative for analytical environment. *Database Systems Journal*, 1(2):3–16, 2010.

[86] Daniel Abadi, Peter Boncz, Stavros Harizopoulos Amiato, Stratos Idreos, and Samuel Madden. *The design and implementation of modern column-oriented database systems*. Now Hanover, Mass., 2013.

[87] Daniel J Abadi, Samuel R Madden, and Nabil Hachem. Column-stores vs. row-stores: how different are they really? In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 967–980, 2008.

[88] Vipin Samar. Unified login with pluggable authentication modules (pam). In *Proceedings of the 3rd ACM conference on Computer and communications security*, pages 1–10, 1996.

[89] Deepa Panse and P Haritha. Multi-factor authentication in cloud computing for data storage security. *International Journal of Advanced Research in Computer Science and Software Engineering*, 4(8):629–634, 2014.

[90] Shyamnath Gollakota, Nabeel Ahmed, Nickolai Zeldovich, and Dina Katabi. Secure in-band wireless pairing. In *USENIX security symposium*, pages 1–16. San Francisco, CA, USA, 2011.

[91] Kiran-Kumar Muniswamy-Reddy, Peter Macko, and Margo I Seltzer. Provenance for the cloud. In *FAST*, volume 10, pages 15–14, 2010.

[92] Peter Buneman, Sanjeev Khanna, and Wang-Chiew Tan. Data provenance: Some basic issues. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 87–93. Springer, 2000.

[93] Boris Glavic, Klaus R Dittrich, A Kemper, H Schöning, T Rose, M Jarke, T Seidl, C Quix, and C Brochhaus. Data provenance: A cctegorization of existing approaches. *BTW'07: Datenbanksysteme in Buisness, Technologie und Web*, (103):227–241, 2007.

[94] Ashish Gehani and Dawood Tariq. Spade: Support for provenance auditing in distributed environments. In *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*, pages 101–120. Springer, 2012.

[95] Yogesh L Simmhan, Beth Plale, and Dennis Gannon. A survey of data provenance techniques. *Computer Science Department, Indiana University, Bloomington IN*, 47405:69, 2005.

[96] Kiran-Kumar Muniswamy-Reddy, David A Holland, Uri Braun, and Margo I Seltzer. Provenance-aware storage systems. In *Usenix annual technical conference, general track*, pages 43–56, 2006.